RESOURCE EFFICIENT INFORMATION INTEGRATION IN CYBER-PHYSICAL SYSTEMS

BY

LU SU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Doctoral Committee:

Professor Tarek F. Abdelzaher, Chair & Director of Research
Professor Jiawei Han
Professor Klara Nahrstedt
Associate Professor Xue Liu, McGill University

Abstract

The proliferation of increasingly capable and affordable sensing devices that pervade every corner of the world has given rise to the fast development and wide deployment of Cyber-Physical Systems (CPS). Hosting a whole spectrum of civilian and military applications, cyber-physical systems have fundamentally changed people's ways of everyday living, working, and interactions with the physical world in general.

Despite their tremendous benefits, cyber-physical systems pose great new research challenges, of which, this thesis targets on one important facet, that is, to understand and optimize the tradeoff between the quality of information (QoI) provided by the sensor nodes and the consumption of system resources. On one hand, individual sensors are not reliable, due to various possible reasons including incomplete observations, environment and circuit board noise, poor sensor quality, lack of sensor calibration, or even intent to deceive. To address this sensor reliability problem, one common approach is to integrate information from multiple sensors that observe the same events, as this will likely cancel out the errors of individual sensors and improve the quality of information. On the other hand, cyber-physical systems usually have limited resources (e.g., energy, bandwidth, storage, time, space, money, devices or even the human labor). Therefore, it is usually prohibitive to collect data from a large number of sensors due to the potential excessive resource consumption.

Targeting on the above challenge, in this thesis we present a suite of resource-efficient information integration tools that can intelligently integrate information from distributed sensors so that the highest quality of information can be achieved, under the constraint of system resources. The proposed information integration toolkit bears superior generalizability and flexibility, and thus can be applied to a full spectrum of application domains.

To Jing and my parents.

Acknowledgments

First and Foremost, I would like to express my deepest gratitude to my thesis advisor, Professor Tarek F. Abdelzaher, for his invaluable guidance, continuous support, and persistent encouragement. I learned tremendously from him in every aspect. His passion in research, teaching, and student supervision has motivated me to pursue a career in academia. He will be my lifetime role model, and I wish I could become a great researcher as well as advisor like him. I feel so fortunate to have him as my advisor.

Next, I would like to thank Professor Jiawei Han for his kindest advice and help on both my research and career. It is my privilege to have chances to work with one of the greatest scholars in data mining research. He is the most important source of inspiration for me to explore research problems at the intersection of data mining and cyber-physical systems. I also sincerely thank my other committee members, Professor Klara Nahrstedt and Professor Xue Liu, for their invaluable inputs. Their constructive suggestions and comments helped me significantly improve my thesis. I am truly honored to have them in my doctoral committee.

During my Ph.D study, I have worked as the teaching assistant of two courses, *Database Systems* and *Introduction to Computing with Application to Engineering and Physical Science*, under the supervision of Professor Kazuhiro Minami and Professor Thomas Gambill, respectively. They were always patient and helpful whenever I had problems. They showed me what it takes to be an excellent teacher and educator. I believe what I learned from them will constantly benefit me in my career.

I would like to extend my gratitude to the professors, researchers, and colleagues who helped me on this thesis and other research projects. In particular, I would like to thank Professor Jennifer Hou, Professor Guohong Cao, Professor P. R. Kumar, Professor Feng Liang, Professor Peng Liu, Yong Yang, Shaohan Hu, Shen Li, Shiguang Wang, Yunlong Gao, Yan Gao, Bolin Ding, Chi

Wang, Changlei Liu, Hui Song, Hengchang Liu, Qi Li for valuable discussions, suggestions, and collaborations.

I have learned a lot through the research experiences in government and industry labs where I got chances to work on real-world problems. Thus, I want to express my special thank to my mentors, managers, and collaborators in IBM T. J. Watson Research Center and National Center for Supercomputing Applications (NCSA), including Dr. Fan Ye, Dr. Seraphin Calo, Dr. Oktay Gunluk, Dr. Ting He, Dr. Yang Song, Dr. Dinesh Verma, Dr. Jian Tan, Dr. Li Zhang, Dr. Mudhakar Srivatsa, Dr. Jim Basney, Dr. Himanshu Khurana, and Dr. Rakesh Bobba.

I want to take this opportunity to thank all of my friends who supported me during these years. It is my good fortune to have so many friends, they make my Ph.D journey a pleasant and exciting one. However, it is impossible to mention all the names. Here I may only be able to list my fellow colleagues in UIUC System and Networking group: Qing Cao, Chengdu Huang, Liqian Luo, Raghu Ganti, Hossein Ahmadi, Md Yusuf Sarwar Uddin, Mohammad Maifi Hasan Khan, Hieu Le, Jin Heo, Praveen Jayachandran, Nam Pham, Eun Soo Seo, Dong Wang, Fatemeh Saremi, Md Tanvir Al Amin, Hongwei Wang, Hongyan Wang, Siyu Gu, Chenji Pan, Fan Yang, Yu-En Tsai, Qixin Wang, Wenbo He, Ying Huang, Zixia Huang, Haiming Jin, Zhenhuan Gao, Hongyang Li, Pengye Xia, Shannon Chen, Wanmin Wu, Long Vu, Debish Fesehaye, Ahsan Arefin, Zheng Zeng, I-Hong Hou, Shu Shi, Yingyi Liang, Xiao Cai, Jianqing Zhang, Ding Yuan, Xiao Ma, Zuoning Yin, Lin Tan, Shan Lu, Weiwei Xiong, Shuo Tang, Haohui Mai, Hui Xue, Steve Ko, Brian Cho, Wenxuan Zhou, Chia-Chi Lin, Chi-Yao Hong, Virajith Jalaparti, Ahmed Khurshid, Fred Douglas, Qingxi Li, Mo Dong, Xuan Zou, Yuhao Zheng, Dong Jin, Qiyan Wang, Shuyi Chen, Farhana Ashraf.

Also, I would like to thank various funding agencies and institutions, such as National Science Foundation, Army Research Laboratory, Office of Naval Research, National Center for Supercomputing Applications (NCSA), and IBM Research, for their financial support and assistance.

Lastly, and most importantly, I wish to thank my father Xiwen Su, my mother Ping Fu, and my wife Jing Gao, for their continuous love and support. They are with me all these years, sharing my pain and happiness. My degree is at the expense of their sacrifices. My family is my most prized possession. To them I dedicate this thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The recent confluence of sensing, computing, and communication technologies has given rise to Cyber-Physical Systems [51, 67, 74], an emergent infrastructure that connects the cyber-world and the physical-world in ways previously unimaginable. Hosting a whole spectrum of civilian and military applications with enormous societal impacts and economic benefits, cyber-physical systems have the potential to push forward the next society-transforming change, just like what the Internet did decades ago.

One major force pushing forward the advances of cyber-physical systems is the proliferation of increasingly capable and affordable sensing devices that have become an integral part of people's everyday lives. Nowadays, applications of cyber-physical systems have fundamentally changed the ways in which we live and work. Examples include high confidence medical devices, assisted living, traffic control and safety, advanced automotive systems, process control, energy conservation, environmental control, avionics, instrumentation, critical infrastructure control (electric power, water resources, and communication systems for example), distributed robotics (telepresence, telemedicine), defense systems, manufacturing, and smart structures [51]. Furthermore, the recent rapid growth of social media enables "people"-centric sensing applications where each participant reports what she learned about the physical and social world from either her own observations or mobile devices. As opposed to the traditional hardware-centric systems, this new people-centric sensing paradigm complements and significantly broadens the application domains of cyber-physical systems.

In a word, the explosion of new applications built upon hardware-centric as well as people-centric cyber-physical systems, though having dramatically improved people's living standard, poses new research challenges, which I discuss in great detail in this thesis.

1.1 Motivation and Challenges

Today, the pervasive hardware-centric as well as people-centric cyber-physical systems are generating data in much larger volume than we are capable of fully consuming. In particular, the challenge confronting us lies in the tradeoff between the quality of information (QoI) provided by the sensors and the consumption of system resources.

- **Quality of Information (QoI):** Individual sensors, hardware- or people-centric, are not reliable. The information provided by them may not accurately represent the real world. The possible reasons include incomplete views of observations, environment and circuit board noise, poor sensor quality, lack of sensor calibration, or even intent to deceive. Inaccurate or even false information could mislead people's decisions, and eventually result in invaluable loss. To address this sensor reliability problem, one common approach is to integrate information from multiple sensors that observe the same events, as this will likely cancel out the errors of individual sensors and improve the quality of information[1].

- **Resource Consumption:** Cyber-physical systems are usually resource-constrained. Resources generally refer to the energy, bandwidth, storage, time, space, money, devices or even the human labor. Overconsumption of resources is generally undesirable due to economic, environmental, and ecological concerns. Therefore, it is usually prohibitive to collect data from a large number of sensors due to the potential excessive resource consumption.

Quality of information and resource consumption are closely intertwined, posing conflicting demands on the design of cyber-physical systems. Intelligently balancing the tradeoff between them is the challenge that I explore in this thesis.

1.2 Thesis Statement

Targeting on the above challenge, I propose the following thesis statement:

By intelligently integrating the information from distributed sensors, we can significantly improve the quality of information under the constraint of system resources.

[1] In this thesis, I assume the majority of sensor nodes are able to provide information with reasonable quality.

1.3 Thesis Overview

In this thesis, towards the objective of making the best use of the available resources to achieve the best quality of information, I present a suite of resource-efficient information integration tools for cyber-physical systems. In this section, I provide an overview of each work and explain how they relate to each other.

1.3.1 Bandwidth Efficient Data Aggregation

Let us start with the simplest information integration scenario, data aggregation [59], which has been put forward as an essential traditional paradigm for routing in wireless sensor network, a representative cyber-physical system. The idea is to use a function (e.g., average, max, and min) to combine the raw sensory data from different sensor nodes enroute to eliminate transmission redundancy and thus save energy as well as bandwidth.

In this work [77], we aim at finding the optimal data collection and transmission rate for each sensor node within a distributed sensor network, such that the throughput (i.e., the amount of data delivered to the sink node) can be maximized, under the constraint of network bandwidth. We initially formulate this rate allocation problem as a network utility maximization problem. But due to its non-convexity, we make a couple of variable substitutions on the original problem and transform it into an approximate problem, which is convex. We then apply duality theory to decompose this approximate problem into a rate control and a scheduling subproblem. Based on this decomposition, a distributed algorithm for joint rate control and scheduling is designed, and proved to approach arbitrarily closely to the optimum of the approximate problem. Finally, we show that our approximate solution achieves near-optimal performance through both theoretical analysis and simulations.

1.3.2 Energy Efficient Decision Aggregation

Data aggregation reduces the amount of sensory data delivered to the base station through combining data packets from multiple sensor nodes into a single packet. However, since in data aggregation each sensor is still required to report its raw data, excessive system resources would be consumed if the raw sensory data collected are complex, high dimensional data such as audio and video clips.

Now the question is, can we do anything better? Can we achieve more data reduction and thus save more resources? The answer is yes! A considerable amount of applications of cyber-physical systems are decision-making-oriented, where a choice is to be made from multiple options based on the information provided by the sensor nodes. For example, a set of acoustic sensors can be deployed on an island by zoologists to identify the species (e.g., sparrow or crow) of the observed birds based on their vocalizations. In this case, there is no need for each sensor to send its raw audio clips to the base station if this sensor can locally process the audio data and figure out the species of the birds it observed. Therefore, we can simply ask each sensor to report its local decision and combine multiple nodes' decisions within the network or at the sink. In this way, we minimize the amount of data a sensor node injects into the network, leading to minimum resource consumption.

This work [76] is carried out around this problem. To solve decision making problems, the most widely used technique is classification [33, 63], which is the task of assigning objects (data) to one of several predefined categories (classes). However, there is a major hurdle in applying classification techniques to solve decision making problems in cyber-physical systems, that is, the limited availability of labeled training data, due to the high cost of manual labeling in the harsh locales where a cyber-physical system is usually deployed. Without sufficiently large training set, classification algorithms tend to make inaccurate predictions.

To address this challenge, we propose a hierarchical aggregation classification (HAC) protocol, which is built on a tree topology where all nodes detect the same events. Instead of employing classification algorithms, HAC lets each sensor node locally conduct cluster analysis, which groups data points only based on the similarity of their feature values without any training. The clustering results of the sensors are then aggregated along the tree through an efficient algorithm called Decision Aggregation which can integrate the limited label information with sensors' clustering results. Eventually, the global agreement is reached at the sink node.

We also design an extended version of HAC, called the constrained hierarchical aggregate classification (cHAC) protocol, which can be easily tuned to adjust the tradeoff between the communication energy and the classification accuracy, and thus adapted to different application performance requirements.

1.3.3 Decision Aggregation with Sensor Selection

In many applications of cyber-physical systems, it is desired that the raw sensory data be delivered as is. For example, zoologists may want to use a cyber-physical system to automatically collect high-quality video or audio records of wild animals [96, 97]. Moreover, some raw data (e.g., video and audio clips) can help people label the events detected by the sensor nodes. However, the system resources may not be able to afford the delivery of all the raw data. Consequently, we have to select sensors to perform data collection and transmission based on the quality of their captured information.

To this end, we provide a quality of information based data selection and transmission service for decision making applications in cyber-physical systems [78]. In this scenario, the QoI of a sensor node bears two aspects, data reliability and data redundancy, both of which can be derived based on the outputs of the decision aggregation procedure.

In particular, reliability implies the degree to which the information provided by a sensor node represents the real world, and can be estimated through exploring the agreement between this node and the majority of others. Redundancy, on the other hand, represents the information overlap among different sensor nodes, and can be measured via investigating the similarity of their clustering results. Based on the proposed QoI metrics, we formulate an optimization problem that aims at maximizing the reliability of sensory data while eliminating their redundancies under the constraint of network resources (e.g., bandwidth). We decompose this problem into a data selection and data transmission subproblem, and develop a distributed algorithm to solve them independently.

1.3.4 Generalized Decision Aggregation

The aforementioned decision aggregation approaches, though yielding reasonably good performance in certain cases, suffer from several limitations, as listed below:

First, they only take as input discrete decision information. Sometimes individual sensors may not be quite confident about their decisions due to various reasons, such as incomplete or noisy observations. In this case, if each sensor's confidence information (i.e., the probability that it "believes" the observed event belongs to each candidate class) can also be taken into consideration,

we should be able to further improve the final decision accuracy.

Second, they assume all the events are observed by all the sensors, and have the same number of classes. In practice, however, different sensor nodes, placed in different locations or having different observation angles, may observe different sets of events. This is especially true for people-centric sensing scenarios where the sensors are human-carried or humans themselves. Moreover, different events may represent different types of objects and thus could have different numbers of candidate classes.

Third, they target mainly on traditional hardware-centric cyber-physical systems where sensory data are collected and processed in a completely unattended manner, and thus very limited labelled training data can be obtained. However, in newly emerged sensing scenarios where people are playing increasingly more critical roles, more opportunities are offered for ground truth label collection. Thus, more flexible strategies are called for to cope with any availability level of label information.

The goal of this work is to develop a *generalized decision aggregation* (GDA) framework for cyber-physical systems that can address the above challenges. The GDA framework is able to take advantage of the confidence information of each sensor about its decision, and thus achieves higher decision accuracy. Targeting generalized problem domains, our framework can naturally handle the scenarios where different sensor nodes observe different sets of events whose numbers of possible classes may also be different. It also makes no assumption about the availability level of ground truth label information, while being able to take advantage of any if present. For these reasons, the proposed GDA framework can be applied to a much broader spectrum of cyber-physical systems.

1.4 Thesis Organization

In summary, the four aforementioned distributed data/decision aggregation strategies form a powerful information integration toolkit that bears superior generalizability and flexibility and meets the requirements of a wide spectrum of application scenarios.

In each of the next four chapters, I will elaborate on a specific information integration tool, shedding light on its design philosophy and application domain. Specifically,

- In Chapter 2, we propose a bandwidth efficient data aggregation scheme that can achieve near-optimal network throughput. This scheme is designed for the applications where only simple statistics (e.g., average, max, and min) of the sensory readings are needed.

- In Chapter 3, to serve decision making applications where the task is to identify the categories (classes) of the observed events or targets, we introduce an energy efficient decision aggregation protocol that can integrate the limited label information with individual sensors' decisions to reach the global agreement.

- In Chapter 4, to deal with the applications where raw data are required to be delivered as is, we provide a quality of information based data selection and transmission service for decision making applications in cyber-physical systems.

- In Chapter 5, a generalized decision aggregation framework is presented. Compared to the previously discussed approaches, this framework is applicable to more general application scenarios, in that i) sensor nodes can provide confidence information on their observed events, ii) different sensor nodes observe different sets of events whose numbers of possible classes may also be different, and iii) the system can dynamically take advantage of any ground truth label availability level.

Finally, Chapter 6 concludes this thesis and provides a discussion of future research.

Chapter 2

Bandwidth Efficient Data Aggregation

2.1 Introduction

Data aggregation [59] has been put forward as an essential paradigm for routing in wireless sensor network, a representative cyber-physical system. The idea is to use a function like average, max or min to combine the data coming from different sources enroute to eliminate transmission redundancy and thus save energy as well as bandwidth. In recent years, a large spectrum of studies have been conducted on various problems of data aggregation in sensor networks. However, the following fundamental question has not been answered thus far: "*Does there exist an optimal rate allocation for data aggregation which maximizes the utilization of network resources and meanwhile maintains certain fairness among all the sources?*". This chapter gives the answer to this question.

Finding the optimal rate allocation for data aggregation in sensor networks can be regarded as a utility-based resource allocation problem. In particular, each source is associated with a utility, which is defined as a function of the source's sending rate. The function value can be conceptually regarded as the quality of information provided by the source. For a given aggregation tree, there exists a unique "maximum utility" rate allocation, at which the network resource utilization is optimal. Meanwhile, certain fairness such as max-min fairness and proportional fairness can be achieved when we choose appropriate utility functions [18].

The problem of maximizing the network utilities has been explored in the context of both wired [18,43,46,47,56] and wireless [12,23,53,54] networks, for rate control in unicast or multicast. Although using a similar approach, we show that rate allocation for data aggregation in wireless sensor networks faces unique challenges both theoretically and practically, making this problem a completely different one to which none of the existing solutions can be applied.

Challenge I: Theoretically, rate allocation for data aggregation is not only subject to the network

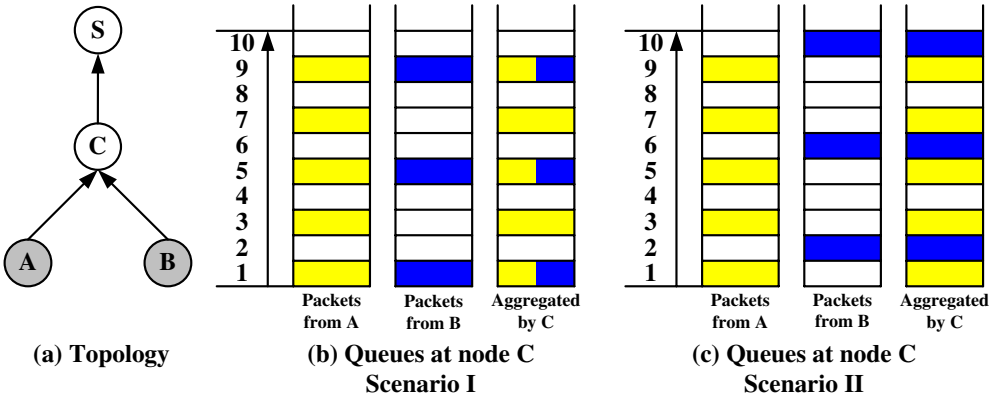


Figure 2.1: An example of data aggregation constraint

capacity constraint, but also the *data aggregation constraint* on the *aggregation nodes* (i.e., non-leaf nodes) of the aggregation tree.

Figure 2.1 provides an intuitive example of the data aggregation constraint. A simple aggregation tree is shown in Fig. 2.1(a). In this case, two source nodes A and B collect and transmit data to node C, who aggregates the received data and forwards them to the sink S. Fig. 2.1(b) and (c) illustrate two different scenarios of data aggregation. In either scenario, three columns are displayed, which correspond to three queues maintained by node C. The first two queues store the packets coming from A and B, while the last one (queue C) keeps the packets generated by aggregating A and B's packets. The packets in each of the three queues are sorted by the timestamps recorded in their headers. In Fig. 2.1(b) and (c), the vertical axis denotes the timestamp, which indicates the time when the carried data packet is collected. In this chapter, we assume that *only the data collected at the same time can be aggregated*. This assumption is valid in many applications of sensor networks, such as target localization and fire alarming, since the data collected at the same time usually contain the information about the same event. For this reason, an aggregated packet has the same timestamp as the raw packets involved in its aggregation. Sometimes, a packet coming from a source node has no coincident packets with the same timestamp from other source nodes, such as the packets with timestamp 3 and 7 in queue A. In this case, the aggregation node does nothing but simply forwards it upwards.

In the first scenario shown in Fig. 2.1(b), the number of packets stored in queue C is the same as the number of packets in queue A, since the time slots when node B collects data are the subset

of the time slots when A collects data. Therefore, to keep the network stable, in other words, to prevent the queue of node C from overflow, the transmission rate of node C should be no less than A's rate. However, this doesn't hold in the second scenario displayed in Fig. 2.1(c), where the only difference from scenario I is that the timestamps of all the packets in queue B are increased by one which implies that node B postpones all its data collections by one time slot. Surprisingly, this causes a fundamental change in queue C. In particular, no aggregation can be made, since there is no coincident packets of A and B. As a result, the number of packets in queue C is the summation of queue A and B's packets. Therefore, in this scenario, the requirement of stability becomes that C should send faster than the aggregate rate of A and B.

This example reveals the fact that the transmission rate of an aggregation node is constrained by not only the rates of its children but also their *packet timestamp patterns*. The packet timestamp pattern of a node includes two components: the intervals between the timestamps of consecutive packets and the time-offsets of the packets among the nodes who share the same parent.

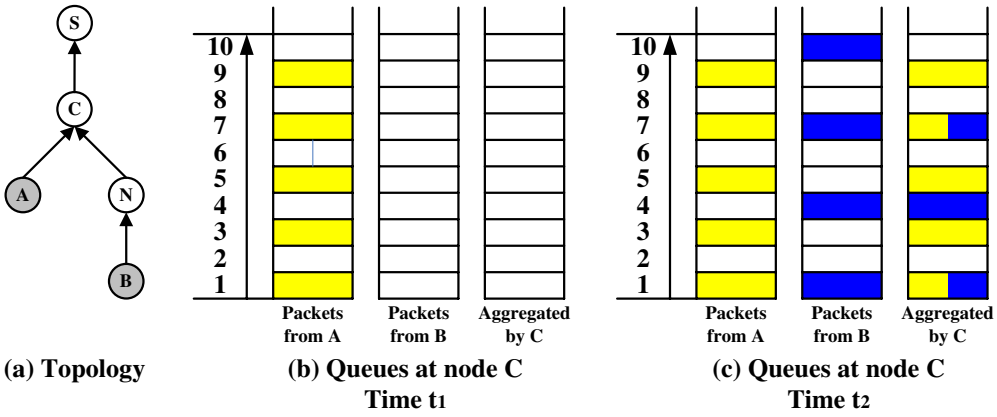


Figure 2.2: An example of data availability constraint

Challenge II: Practically, rate allocation for data aggregation has an implicit constraint, which is referred to as the *data availability constraint* in this chapter. Figure 2.2 gives us an illustrative example of this constraint. Similar to the previous example, node A and B work as the source nodes. However, B is not directly connected to the aggregation node C. An intermediate node N relays data for B. Suppose at time t_1, as shown in Fig. 2.2(b), A has delivered some data to C, whereas B's data has not arrived at C since they are delayed at node N. At this moment, although lots of A's packets are waiting in its buffer, node C needs to wait until B's data arrives at time t_2

(for the sake of simplicity, suppose A transmits no data during this period), and then fulfills its aggregation task, as shown in Fig. 2.2(c). This is because if C chooses to deliver A's packets at time t_1, it has to do the same job again when B's packets arrive. As a result, double amount of traffic is injected into the downstream links by C. This is definitely not an economic solution from the perspective of network utility maximization.

The main purpose of our work is to address the above challenges. We first formulate this rate allocation problem as a network utility maximization problem. Due to its non-convexity, we take a couple of variable substitutions on the original problem and transform it into an approximate problem, which is convex. We then apply duality theory to decompose this approximate problem vertically into a rate control subproblem and a scheduling subproblem that interact through shadow prices. Based on this decomposition, a distributed subgradient algorithm for joint rate control and scheduling is designed, and proved to approach arbitrarily close to the optimum of the approximate problem. Finally, we show that our approximate solution can achieve near-optimal performance through both theoretical analysis and simulations. To the best of our knowledge, this work is the first one to present a joint design of rate allocation and scheduling in the context of data aggregation in wireless sensor networks.

The rest of the chapter is organized as follows. We introduce the system model in Section 2.2 and formulate the problem in Section 2.3. In Section 2.4, the original problem is transformed into a convex approximate problem, with the solution given in Section 2.5. In Section 2.6, we explain how the proposed solution is implemented in a decentralized manner. Then, we discuss some related issues in Section 2.7, and evaluate the performance of the proposed schemes in Section 2.8. Section 2.9 summarizes the related work. Section 2.10 concludes the chapter.

2.2 System Model

In this section, we explain in detail the system model.

2.2.1 Aggregation Model

We consider an aggregation tree T rooted at the sink. We denote the set of tree edges (links) by $\mathcal{L} = L(T)$. The sensor nodes on T can be divided into *source nodes* that collect sensory readings,

and *aggregation nodes* that aggregate and relay sensory readings. In the rest of the chapter, we assume that the source nodes are only at the leaf nodes of the aggregation tree. However, it is possible that a sensor node plays a dual role as both source node and aggregation node. This problem can be easily addressed through a simple graph transformation. In particular, we replace each source node at a non-leaf node by creating a new leaf node and placing the source node in it, and then connect the new leaf node to the non-leaf node where the source node is originally located by a link with infinite capacity[1].

As in most applications of sensor networks, we assume that all the sensor nodes are equipped with the same sensing and communicating devices, and furthermore, the maximum collecting rate of the sensing device is larger than the maximum transmitting rate of the communicating device. The lifetime of the network is divided into slots with equal duration, and each slot is further divided into subslots. The sensing device of each node may or may not collect data at a subslot. Once it decides to work within a subslot, it always collects data at its maximum speed. The data collected within a subslot is encapsulated into a packet, which is referred to as the *Basic Transmission Unit (BTU)*.

2.2.2 Probabilistic Rate Model

As discussed in Section 2.1, the transmission rate of an aggregation node is constrained by not only the rates of its children but also their packet timestamp patterns. To characterize the packet timestamp patterns of sensor nodes, we introduce a *probabilistic rate model*, which is defined and explained as below:

Definition 1. Probabilistic Rate Model. At each subslot, the sensing device of a node chooses to collect data or not, based on a probability, which is referred to as the *data collection rate* of this node.

Suppose a time slot is composed of 100 subslots, and the data collection rate of a given sensor node is 0.5. Then, it works at roughly 50 randomly selected subslots and sleeps at the rest of time. As a result, it collects around 50 packets (BTU) within a time slot.

[1] In practice, we do not place this link in any independent set, and thus it has no impact on the scheduling problem.

We define the *data transmission rate* of a node as the ratio of the number of packets (BTU) this node delivers within a time slot to the number of subslots in each slot. Data transmission rate is actually a normalized rate. Its relation with the genuine transmission rate, which is equal to the number of bits that a node delivers within a time slot, can be reflected by the formula: $x^N = \frac{x^G}{C_B \times N_{sub}}$, where x^N and x^G denote the normalized and genuine data transmission rate, respectively. In addition, C_B is the size of BTU, and N_{sub} represents the number of subslots in each slot. For example, assume the data transmission rate of a node is 1000bps, C_B is 100bits and N_{sub} is 20. Based on the above formula, the normalized data transmission rate is 0.5.

The probabilistic rate model can be considered as a generalization of a node's packet timestamp pattern. This can be better understood after we mathematically formulate the problem in the next section. Additionally, in the sensing tasks which require periodic data collection, the probabilistic rate model can capture the long-term expectation of time-varying time-offsets of the packets from different nodes. It is extremely difficult to mathematically model the fixed time-offsets. More detailed discussion can be found in Section 2.7.2.

2.3 Problem Formulation

2.3.1 Terminology

Link flow: We define link flow as the single-hop data traffic going through a link (tree edge) of the aggregation tree. Conceptually, it includes *source flow* originating at source nodes and *aggregation flow* coming from aggregation nodes. The set of source flows and the set of aggregation flows are denoted by F^S and F^A. In addition, $\mathcal{F} = F^S \bigcup F^A$ represents the set of all the flows on the aggregation tree. For any flow $f \in \mathcal{F}$, we use $\pi(f)$ and $C(f)$ to denote its parent flow and set of children flows, respectively. Finally, $f(l)$ implies the flow which goes through a link $l \in \mathcal{L}$, and $l(f)$ means the link through which a flow $f \in \mathcal{F}$ passes. We denote the normalized rate of each flow $f \in \mathcal{F}$ by x_f. For a source flow $f \in F^S$, its rate is quantitatively equal to the data collection rate of the corresponding source node. We use X_f to denote the interval in which x_f must lie:

$$X_f := \{[m_f, M_f] \quad \text{if } f \in F^S; \quad [0, M_f] \quad \text{if } f \in F^A\}.$$

where m_f and M_f are the lower and upper bounds of x_f.

Queue: At each aggregation node, the packets coming from each of its children flows f is buffered in a queue, denoted by $Q(f)$. The packets in each queue are sorted by their timestamps, as shown in Fig. 2.1 and Fig. 2.2. We use $T_t(f)$ to denote the timestamp of the packet on the top of $Q(f)$ (largest timestamp), and $T_b(f)$ to denote the timestamp of the packet at the bottom of $Q(f)$ (smallest timestamp). In addition, the aggregated packets (available data) are stored in a separate queue, waiting for transmission.

2.3.2 Constraints

In this part, we elaborate on the constraints that our objective function is subject to.

Network Capacity Constraint: Based on the network topology, a conflict graph [41] can be constructed to capture the contention relations among the links. In the conflict graph, each vertex represents a link, and an edge between two vertices implies the contention between the two corresponding links, i.e., they cannot transmit at the same time. Given a conflict graph, we can identify all its independent sets of vertices. The links in an independent set can transmit simultaneously.

Let $\mathcal{I}$ denote the set of independent sets. We represent an independent set, I_i $(i = 1, 2, ..., |\mathcal{I}|)$, as a $|\mathcal{L}|$-dimensional rate vector, which is r^i. In r^i, the lth entry is $r^i_l := \{c_l \quad \text{if } l \in I_i; \quad 0 \quad \text{otherwise}\}$ where c_l denotes the capacity of link $l \in \mathcal{L}$. Here we should note that this capacity is a normalized capacity, which is defined as the ratio of the maximum number of packets (BTU) that can be delivered through a link within a time slot to the number of subslots in each slot. The feasible capacity region Π at the link layer is defined as the convex hull of these rate vectors:

$$\Pi := \{r \mid r = \sum_{i=1}^{|\mathcal{I}|} \alpha_i r^i,\ \alpha_i \geq 0,\ \sum_{i=1}^{|\mathcal{I}|} \alpha_i = 1\}.$$

With above notations, now we can formally define the *network capacity constraint* as follows:

$$x_{f(l)} \leq r_l \qquad \text{for all } \ l \in \mathcal{L}.$$

Namely, the rate of each flow must not exceed the amount of capacity allocated to the link it passes through. Let $r := \{(r_{l_1}, r_{l_2}, ..., r_{l_{|\mathcal{L}|}}) |\ \ l_i \in \mathcal{L}\}$, and it should satisfy $r \in \Pi$.

Data Aggregation Constraint: As in the rate control of unicast and multicast scenarios, it is essential to investigate the relationship between the rate of a parent flow and the rates of its children flows so as to stabilize the network. However, the difficulty of achieving this goal in the context of data aggregation is much larger, since a slight change in the packet timestamp pattern of a node may incur significant change in the resulting aggregated packets, as disclosed in Section 2.1. To overcome this difficulty, we adopt a probabilistic rate model, which is defined in Section 2.2. Given the rate of a node, this model can generalize all the possibilities of this node's packet timestamp patterns.

Under the probabilistic rate model, the *data aggregation constraint* can be formulated as follows:

$$1 - \prod_{f_c \in C(f)} (1 - x_{f_c}) \leq x_f \qquad \text{for all} \quad f \in F^A.$$

Here we give an interpretation of this constraint which may draw a better understanding. We say a node covers a subslot if there exists a packet transmitted by this node whose timestamp is exactly this subslot. Then, $1 - x_f$ denotes the percentage of the subslots which are not covered by the sending node of f. Later, we use the concepts of a flow and its sending node interchangeably. It follows that $\prod_{f_c \in C(f)}(1 - x_{f_c})$ implies the percentage of the subslots not covered by any of f's children nodes. Intuitively, a parent node needs to cover all the subslots covered by at least one child, which is $1 - \prod_{f_c \in C(f)}(1 - x_{f_c})$. By this intuition, the data aggregation constraint is presented.

Data Availability Constraint: From the example shown in Fig. 2.2, we learn that an aggregation node cannot take any actions before it makes sure that all the packets collected at the same time have arrived. Given a timestamp, after receiving a packet with this timestamp or larger timestamp from each child node, the aggregation node will know that all the packets with this timestamp have arrived. Here we assume that packets arrive in the order of their timestamps. Then, it performs the aggregation and puts the resulting aggregated packet into the queue where the packets available for transmission are stored. Thus, the timestamps of the available packets are constrained within the time interval from the smallest $T_b(f)$ to the smallest $T_t(f)$ among all the queues. Recall that $T_b(f)$ and $T_t(f)$ denote the timestamps of the packets on the bottom and the top of each queue, respectively.

With a binary indicator variable $b_\tau(f)$ defined as below:

$$b_\tau(f) = \begin{cases} 1 & \text{There is a packet with timestamp } \tau \text{ in } Q(f). \\ 0 & \text{otherwise.} \end{cases}$$

In terms of the number of BTUs, we denote the amount of the available data at the sending node of f by λ_f. λ_f can be calculated as follows:

$$\lambda_f = \sum_{\tau=\min_{f_i \in C(f)} T_b(f_i)}^{\min_{f_j \in C(f)} T_t(f_j)} \left(\bigvee_{f_k \in C(f)} b_\tau(f_k) \right) \tag{2.1}$$

where $\bigvee$ denotes the bitwise operation "OR".

By this formula, we can easily check that in the example shown in Fig. 2.2, the amount of the available data at node C (stored in queue C) at time t_1 (Fig. 2.2(b)) and t_2 (Fig. 2.2(c)) are 0 and 6, respectively. Note that in the scenario happens at t_2, we do not take into account the packet with timestamp 10 in queue B, since it is still unknown at this moment whether node A also has a packet with timestamp 10, and thus cannot mark this packet to be available.

Furthermore, let a_f be the amount of data which can be transmitted by the sending node of f. Then, the *data availability constraint* can be formally defined as follows:

$$a_f \leq \lambda_f \qquad \text{for all} \quad f \in F^A.$$

In other words, for each flow $f \in F^A$, it can not deliver more data than the amount of the available data stored at its sending node. Within a time slot, once the available data are all sent out, the sending node can not do more transmission even if some of the queues for its children nodes are not empty. The data availability constraint minimizes the amount of packets a node could inject into the network. By this constraint, for each timestamp, there is at most one packet with this timestamp arriving at the sink. More importantly, the data availability constraint is actually the prerequisite of the data aggregation constraint, since the data aggregation constraint implicitly assumes all the packets from different sources collected at the same time are merged into a single packet.

2.3.3 Problem Formulation

With the terminologies and constraints defined above, we can now formulate the problem to be solved. We associate each source flow $f \in F^S$ with a utility function $U_f(x_f) : \mathbf{R}_+ \to \mathbf{R}_+$. In this chapter, we assume U_f is continuously differentiable, increasing, and strictly concave. Our objective is to choose the rate of each flow x_f and the allocated capacity of each link r_l so as to maximize the aggregate utility function. We now formulate the problem of optimal rate allocation for data aggregation in sensor networks as the following constrained nonlinear optimization problem:

$$\mathbf{P}: \max \quad \sum_{f \in F^S} U_f(x_f) \tag{2.2}$$

$$\text{subject to} \quad x_{f(l)} \le r_l \qquad \text{for all } l \in \mathcal{L} \tag{2.3}$$

$$a_f \le \lambda_f \qquad \text{for all } f \in F^A \tag{2.4}$$

$$1 - \prod_{f_c \in C(f)} (1 - x_{f_c}) \le x_f \qquad \text{for all } f \in F^A \tag{2.5}$$

$$x \in X, \quad r \in \Pi$$

In **P**, the data availability constraint (2.4) works as the prerequisite of the data aggregation constraint (2.5). However, it is actually an implicit constraint that need not be considered when solving this optimization problem, although in practice each aggregation node works following this constraint. We will give more detailed explanation on this point in Section 2.7.1.

By choosing appropriate utility functions, the optimal rate allocation can achieve different fairness models among the flows [46, 47]. For instance, if we let $U_f(x_f) = w_f \ln(x_f)$ for $f \in F^S$, the weighted proportional fairness can be achieved.

2.4 Approximate Problem

2.4.1 Variable Substitution

Though we formulate the problem of optimal rate allocation, it turns out to be a non-convex program, due to the non-convexity of the data aggregation constraint (2.5). To address this problem, we reorganize the data aggregation constraint and take a log transform on both sides:

$\ln(1-x_f) \leq \sum_{f_c \in C(f)} \ln(1-x_{f_c})$. Next, we substitute x_f of each flow by $\tilde{x}_f = -\ln(1-x_f)$, where we call $\tilde{x}_f$ the *transformed rate* of f. By this variable substitution, the data aggregation constraint becomes: $\sum_{f_c \in C(f)} \tilde{x}_{f_c} \leq \tilde{x}_f$. In the rest of this chapter, we refer to this constraint as the *transformed aggregation constraint*. In addition, based on the feasible region of x_f ($f \in \mathcal{F}$), we can derive the feasible region of $\tilde{x}_f$ as

$$\widetilde{X}_f := \begin{cases} [-\ln(1-m_f), -\ln(1-M_f)] & f \in F^S \\ [0, -\ln(1-M_f)] & f \in F^A \end{cases}$$

where $-\ln(1-m_f) \geq 0$ and $-\ln(1-M_f) < \infty$.

By the variable substitution described above, we transform the data aggregation constraint into a linear constraint. However, this substitution has a side effect, namely, it turns the network capacity constraint into $1 - \exp(-\tilde{x}_{f(l)}) - r_l \leq 0$, another non-convex constraint. To overcome this problem, we reorganize this constraint and take a log transform on both sides, then we have $\tilde{x}_{f(l)} \leq -\ln(1-r_l)$. Next, we take another variable substitution on r_l: $\tilde{r}_l = -\ln(1-r_l)$ where we call $\tilde{r}_l$ the *transformed allocated capacity* of link $l \in \mathcal{L}$. By this variable substitution, the non-convex constraint is transformed into $\tilde{x}_{f(l)} - \tilde{r}_l \leq 0$. We name this constraint as the *transformed capacity constraint*. Recall that r_l is a normalized capacity allocated to link l, and thus it satisfies $0 \leq r_l \leq 1$. Since $r_l = 1 - \exp(-\tilde{r}_l)$, it can be derived that $0 \leq \tilde{r}_l \leq \infty$. By substituting $\tilde{r}_l$ for r_l, the original capacity region Π is transformed into a *transformed capacity region* Π':

$$\Pi' := \{\tilde{r} \mid \tilde{r}_{l_i} = -\ln(1-r_{l_i}),\ i = 1, 2, ..., |\mathcal{L}|,\ r \in \Pi\}$$

However, Π' is not a convex region. Figure. 2.3 illustrates this region transformation. Particularly, Fig. 2.3(a) shows an example of two-dimensional capacity region Π, and the transformed capacity region Π' is drawn in Fig. 2.3(b). As can be seen, Π' (the shaded area) is not a convex region.

To tackle this problem, we constitute an *approximate transformed capacity region* $\widetilde{\Pi}$, which is convex. Recall that the original capacity region Π is actually the convex hull of the rate vectors of all the independent sets. In fact, these rate vectors are the extreme points of Π, since each of

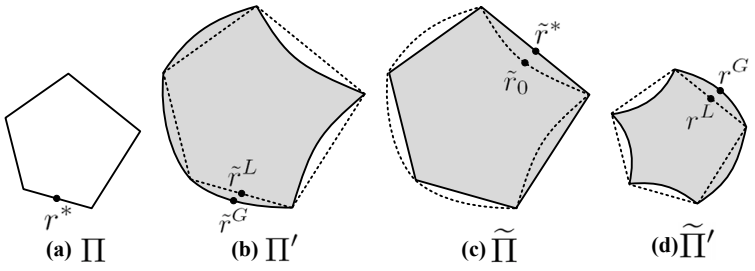


Figure 2.3: Region transformation

them cannot be represented by the convex combination of others. In the transformed capacity region Π', let $\tilde{r}^i$ denote the point (vector) transformed from the ith extreme point r^i (rate vector of ith independent set) in Π. In $\tilde{r}^i$, the lth entry is $\tilde{r}^i_l := \{\tilde{c}_l \quad \text{if} \quad l \in I_i; \quad 0 \quad \text{otherwise}\}$ where $\tilde{c}_l$ is referred to as the *transformed capacity*, and defined by $\tilde{c}_l = -\ln(1 - c_l)$.

Now, we can define the approximate transformed capacity region $\widetilde{\Pi}$ as the convex hull of these transformed rate vectors:

$$\widetilde{\Pi} := \{\tilde{r} \mid \tilde{r} = \sum_{i=1}^{|\mathcal{I}|} \alpha_i \tilde{r}^i, \ \alpha_i \geq 0, \ \sum_{i=1}^{|\mathcal{I}|} \alpha_i = 1\}.$$

It is not difficult to prove that each $\tilde{r}^i, \ i = 1, 2, ..., |\mathcal{I}|$ cannot be represented by the convex combination of others either, and thus is an extreme point of $\widetilde{\Pi}$. Therefore, for each (ith) independent set, there is a one-to-one mapping between its corresponding extreme points in Π (i.e., r^i) and $\widetilde{\Pi}$ (i.e., $\tilde{r}^i$).

Figure 2.3(c) shows the approximate capacity region $\widetilde{\Pi}$ (the shaded area), which corresponds to the original capacity region Π in Fig. 2.3(a). As can be seen, despite of the convexity it achieves, it does not cover all the points of the transformed capacity region Π' (the area enclosed by the dashed curve). Furthermore, it includes some points outside the boundary of Π', and this implies that $\widetilde{\Pi}$ may result in some solutions which are not feasible in the original problem. Actually, if we take a reverse variable substitution (i.e., $r_l = 1 - \exp(-\tilde{r}_l)$) on each point $\tilde{r} \in \widetilde{\Pi}$, a new region denoted by $\widetilde{\Pi}'$ is attained, and shown in Fig. 2.3(d) (the shaded area). As one can see, it does have some points outside the original capacity region Π (the area enclosed by the dashed curve).

However, in our algorithm that will be introduced in the next section, we do not map the solution in $\widetilde{\Pi}$ to Π in this way, namely, through $r_l = 1 - \exp(-\tilde{r}_l)$. Instead, we design a safe mapping scheme,

which can guarantee that there always exists a feasible point in Π, which corresponds to the solution attained in the context of $\widetilde{\Pi}$.

Based on the definition of $\widetilde{\Pi}$, any point in $\widetilde{\Pi}$, say $\tilde{r}_0$, can be expressed as $\tilde{r}_0 = \sum_{i=1}^{|\mathcal{I}|} \alpha_i \tilde{r}^i$. Its counterpart in the original problem is $r^G := \{(r_l^G)|\ r_l^G = 1 - \exp(-\tilde{r}_{0l}),\ l \in \mathcal{L}\}$, and it may not be located inside Π. However, in Π, we can always find a point, which is $r^L = \sum_{i=1}^{|\mathcal{I}|} \alpha_i r^i$ where each α_i equals the α_i in $\tilde{r}_0 = \sum_{i=1}^{|\mathcal{I}|} \alpha_i \tilde{r}^i$. By this mapping scheme, wherever the optimal solution is located in $\widetilde{\Pi}$, our algorithm can identify a corresponding feasible solution inside Π. In the rest of this chapter, we refer to r^G as the *genuine mapping point* of $\tilde{r}_0$, and r^L as the *linear mapping point* of $\tilde{r}_0$. Similarly, given a point $r_0 = \sum_{i=1}^{|\mathcal{I}|} \alpha_i r^i$ inside Π, we can define $\tilde{r}^G := \{(\tilde{r}_l^G)|\ \tilde{r}_l^G = -\ln(1 - r_{0l}),\ l \in \mathcal{L}\}$ and $\tilde{r}^L = \sum_{i=1}^{|\mathcal{I}|} \alpha_i \tilde{r}^i$ as the *genuine mapping point* and the *linear mapping point* of r_0 in $\widetilde{\Pi}$.

Now, we can formally define the *approximate problem* as follows:

$$\widetilde{\mathbf{P}}:\ \max \quad \sum_{f \in F^S} U_f(1 - \exp(-\tilde{x}_f)) \tag{2.6}$$

$$\text{subject to} \quad \tilde{x}_{f(l)} - \tilde{r}_l \leq 0 \qquad \text{for all } l \in \mathcal{L} \tag{2.7}$$

$$\sum_{f_c \in C(f)} \tilde{x}_{f_c} - \tilde{x}_f \leq 0 \qquad \text{for all } f \in F^A \tag{2.8}$$

$$\tilde{x} \in \widetilde{X}, \quad \tilde{r} \in \widetilde{\Pi}$$

According to [6] (Chapter 3.2.4), since $1 - \exp(-\tilde{x}_f)$ is a strictly concave and increasing function, the objective function (2.6) remains strictly concave and increasing. Thus, $\widetilde{\mathbf{P}}$ is a convex problem, and always has a unique maximizer. Once we identify this maximizer, we can use its linear mapping point in Π as the *approximate solution* of $\mathbf{P}$.

2.4.2 Approximation Analysis

In this subsection, we provide some theoretical analysis on both the original problem $\mathbf{P}$ and the approximate problem $\widetilde{\mathbf{P}}$.

Theorem 1. The optimal solution of $\mathbf{P}$ ($\widetilde{\mathbf{P}}$) must be attained on the boundary of Π ($\widetilde{\Pi}$).

Proof. Here we only show the proof for $\mathbf{P}$, since the proof for $\widetilde{\mathbf{P}}$ is similar. By contradiction,

suppose the optimal solution of $\mathbf{P}$, denoted by r^*, is a strictly interior point of Π[2]. Since in Π, the components of r^* only appear in the network capacity constraint (i.e., $x_{f(l)} \leq r_l$), we do not need to check other constraints. At optimality, the network capacity constraint may or may not be active (we say a constraint is active if it attains equality). If it is not active, $x_{f(l)}$ will not change if we increase r_l. On the other hand, if it is active, $x_{f(l)}$ will go up to some extent with the increase of r_l. As a result, the objective value will be improved, since it's strictly increasing with $x_{f(l)}$. Since r^* is an interior point, there must exist some room to increase some components of r^*, without changing the others. This conflicts the assumption that r^* is the optimal solution. Therefore, r^* must be located on the boundary of Π. □

In Π ($\widetilde{\Pi}$), which is a compact $|\mathcal{L}|$-dimensional polyhedron, each facet of its boundary is defined by the convex hull of $|\mathcal{L}|$ extreme points. Thus, the optimal solution of $\mathbf{P}$ ($\widetilde{\mathbf{P}}$) can be expressed as $r^* = \sum_{i=1}^{|\mathcal{L}|} \alpha_i r^i$ ($\tilde{r}^* = \sum_{i=1}^{|\mathcal{L}|} \alpha_i \tilde{r}^i$). In addition, the approximate solution, which is the linear mapping point of $\tilde{r}^*$, is also located at the boundary of Π.

Now, we are interested in how far our approximate solution is from the optimal solution. In other words, we want to know the difference between our approximate objective value and the optimal objective value. We first introduce some notations. For any point r_0 ($\tilde{r}_0$) in Π ($\widetilde{\Pi}$), we define $\mathbf{P}(r_0)$ ($\widetilde{\mathbf{P}}(\tilde{r}_0)$) as the optimization problem $\mathbf{P}$ ($\widetilde{\mathbf{P}}$) when r ($\tilde{r}$) is fixed to be r_0 ($\tilde{r}_0$). In addition, let $\mathbf{P}^*(r_0)$ ($\widetilde{\mathbf{P}}^*(\tilde{r}_0)$) be the optimal objective value of $\mathbf{P}(r_0)$ ($\widetilde{\mathbf{P}}(\tilde{r}_0)$). Suppose r^* ($\tilde{r}^*$) is the global optimal solution in Π ($\widetilde{\Pi}$), we use $\mathbf{P}^* = \mathbf{P}^*(r^*)$ ($\widetilde{\mathbf{P}}^* = \widetilde{\mathbf{P}}^*(\tilde{r}^*)$) to denote the global optimal objective value of $\mathbf{P}$ ($\widetilde{\mathbf{P}}$).

Then, we investigate the performance of the approximate solution. Suppose the objective value of our approximate solution is $\hat{\mathbf{P}}^*$. In the rest of this section, we first show that the difference between the global optimal objective value of $\mathbf{P}$ (i.e., $\mathbf{P}^*$) and $\hat{\mathbf{P}}^*$ is bounded by $\widetilde{\mathbf{P}}^* - \hat{\mathbf{P}}^*$ through Theorem 2, and then give a looser but simpler bound by Theorem 3.

Theorem 2. The optimal objective value of the original problem $\mathbf{P}$ is upper bounded by the optimal objective value of the approximate problem $\widetilde{\mathbf{P}}$.

Proof. Let the point in Π which maximizes $\mathbf{P}$ be r^*, as shown in Fig. 2.3(a). Thus, r^* can be expressed as $r^* = \sum_{i=1}^{|\mathcal{L}|} \alpha_i r^i$. Suppose its genuine mapping point in $\widetilde{\Pi}$ is $\tilde{r}^G$. As can be seen in

[2]In fact, a solution also includes the rate x, here we only consider the capacity r simply for the ease of expression.

Fig. 2.3(b), it may not be inside $\widetilde{\Pi}$. However, in $\widetilde{\Pi}$, we can always find the linear mapping point of r^*, which is denoted by $\tilde{r}^L$ and shown in Fig. 2.3(b). Since the function $f(x) = -\ln(1-x)$ is strictly convex, it can be derived that for each $l \in \mathcal{L}$, $\tilde{r}_l^G = -\ln(1-r_l^*) \leq \sum_{i=1}^{|\mathcal{L}|} \alpha_i(-\ln(1-r_l^i)) = \sum_{i=1}^{|\mathcal{L}|} \alpha_i \tilde{r}_l^i = \tilde{r}_l^L$. Similar to the proof of Theorem 1, we can show that $\widetilde{\mathbf{P}}^*(\tilde{r}^G) \leq \widetilde{\mathbf{P}}^*(\tilde{r}^L)$ by moving each component of $\tilde{r}^G$ towards $\tilde{r}^L$. Since $\mathbf{P}^* = \widetilde{\mathbf{P}}^*(\tilde{r}^G)$ and $\widetilde{\mathbf{P}}^*(\tilde{r}^L) \leq \widetilde{\mathbf{P}}^*$, it can be concluded that $\mathbf{P}^* \leq \widetilde{\mathbf{P}}^*$. □

By Theorem 2, the approximation ratio of our solution can be bounded by $\frac{\widetilde{\mathbf{P}}^* - \hat{\mathbf{P}}^*}{\hat{\mathbf{P}}^*}$. Next, we give a looser but simpler bound of $\mathbf{P}^* - \hat{\mathbf{P}}^*$.

Theorem 3. Suppose that the optimal solution of $\widetilde{\mathbf{P}}$ is $\tilde{r}^*$, and its linear mapping point in Π, i.e., the approximate solution is r^L. Furthermore, let $\tilde{r}_0$ be r^L's corresponding genuine mapping point in $\widetilde{\Pi}$. Then, the value of $\mathbf{P}^* - \hat{\mathbf{P}}^*$ is bounded by $\mu^{\alpha *T}(\tilde{r}^* - \tilde{r}_0)$, where $\mu^{\alpha *}$ represents the vector of the optimal dual variables of problem $\widetilde{\mathbf{P}}(\tilde{r}_0)$.

Proof. As can be seen, $\tilde{r}^*$ and $\tilde{r}_0$ are shown in Fig. 2.3(c), and r^L is shown in Fig. 2.3(d). Since r^L is the approximate solution, by Theorem 2, $\mathbf{P}^* - \mathbf{P}^*(r^L)$ is bounded by $\widetilde{\mathbf{P}}^*(\tilde{r}^*) - \mathbf{P}^*(r^L)$, which is further equal to $\widetilde{\mathbf{P}}^*(\tilde{r}^*) - \widetilde{\mathbf{P}}^*(\tilde{r}_0)$. Based on the theory of perturbation and sensitivity (Chapter 5.6 in [6]), we denote the perturbed version of the optimization problem $\widetilde{\mathbf{P}}(\tilde{r}_0)$ by $\widetilde{\mathbf{P}}_{\tilde{\mathbf{r}}_\mathbf{0}}$, in which the transformed capacity constraint is replaced by $\tilde{x}_{f(l)} - \tilde{r}_{0l} \leq u_l$. Here $u := (u_l, l \in \mathcal{L})$ is the vector of perturbation variables. It is evident that $\widetilde{\mathbf{P}}_{\tilde{\mathbf{r}}_\mathbf{0}}$ coincides with problem $\widetilde{\mathbf{P}}(\tilde{r}_0)$ when u is a zero vector. On the other hand, when u_l is positive it means that we have relaxed the transformed capacity constraint of link l.

We denote the optimal objective value of $\widetilde{\mathbf{P}}_{\tilde{\mathbf{r}}_\mathbf{0}}$ at u by $\widetilde{\mathbf{P}}^*_{\tilde{\mathbf{r}}_\mathbf{0}}(u)$. According to [6] (Chapter 5.6.1), since problem $\widetilde{\mathbf{P}}(\tilde{r}_0)$ is concave, $\widetilde{\mathbf{P}}^*_{\tilde{\mathbf{r}}_\mathbf{0}}(u)$ is a concave function of u. It follows that $\widetilde{\mathbf{P}}^*_{\tilde{\mathbf{r}}_\mathbf{0}}(u) \leq \widetilde{\mathbf{P}}^*_{\tilde{\mathbf{r}}_\mathbf{0}}(0) + \mu^{\alpha *T} u$. Therefore, let $u = \tilde{r}^* - \tilde{r}_0$, the difference between $\mathbf{P}^*$ and $\mathbf{P}^*(r^L)$ can be bounded as follows: $\mathbf{P}^* - \mathbf{P}^*(r^L) \leq \widetilde{\mathbf{P}}^*(\tilde{r}^*) - \widetilde{\mathbf{P}}^*(\tilde{r}_0) = \widetilde{\mathbf{P}}^*_{\tilde{\mathbf{r}}_\mathbf{0}}(u) - \widetilde{\mathbf{P}}^*_{\tilde{\mathbf{r}}_\mathbf{0}}(0) \leq \mu^{\alpha *T} u = \mu^{\alpha *T}(\tilde{r}^* - \tilde{r}_0)$.

Let $f(x) = -\ln(1-x)$, and thus $f^{-1}(y) = 1 - \exp(-y)$. As previously discussed, $\tilde{r}^* = \sum_{i=1}^{|\mathcal{L}|} \alpha_i \tilde{r}^i$. It follows that $r^G = f^{-1}(\sum_{i=1}^{|\mathcal{L}|} \alpha_i \tilde{r}^i)$ (for the sake of simplicity, here we use the notation of a vector to delegate all of its components.) and $r^L = \sum_{i=1}^{|\mathcal{L}|} \alpha_i r^i$. Similar to the proof of Theorem 2, it can be proved that $r^G \geq r^L$ (i.e., $r_l^G \geq r_l^L$). Since $f(x)$ is strict increasing, and

$\tilde{r}_0 = f(r^L)$, it can be inferred that $\tilde{r}^* \geq \tilde{r}_0$ (i.e., $\tilde{r}^*_l \geq \tilde{r}_{0l}$). Therefore, each component of $\tilde{r}^* - \tilde{r}_0$ is nonnegative. Furthermore, since $\tilde{r}_0 = f(\sum_{i=1}^{|\mathcal{L}|} \alpha_i f^{-1}(\tilde{r}^i))$, $\mu^{\alpha *T}(\tilde{r}^* - \tilde{r}_0)$ is a function of $\mu^{\alpha *}$ and $\tilde{r}^i,\ i = 1, 2, ..., |\mathcal{L}|$. □

From this bound, it can be seen that $\mathbf{P}^* - \hat{\mathbf{P}}^*$ is proportional to the difference between $\tilde{r}^*$ and $\tilde{r}_0$. Actually, it is not difficult to show that when the capacity of each link decreases, the difference between $\tilde{r}^*$ and $\tilde{r}_0$ will drop accordingly. However, this does not necessarily means that $\mu^{\alpha *T}(\tilde{r}^* - \tilde{r}_0)$ will also drop, since $\mu^{\alpha *}$ may increase with the decrease of capacities[3]. In fact, $\mu^{\alpha *}$ depends on the particular utility function we choose, and thus there is no universal conclusion on this point. In Section 2.8, we will show an example in which $\mathbf{P}^* - \hat{\mathbf{P}}^*$ drops when the capacity of each link is reduced.

2.5 Cross Layer Design via Dual Decomposition

2.5.1 The Dual Problem

Solving $\widetilde{\mathbf{P}}$ directly requires global coordination of all flows, which is impractical in a distributed environment such as sensor networks. Since $\widetilde{\mathbf{P}}$ is a convex program with compact feasible region, strong duality can be achieved[4] (Chapter 5.2.3 in [6]). Therefore, there exists a unique maximizer $(\tilde{x}^*, \tilde{r}^*)$ for $\widetilde{\mathbf{P}}$, which can be attained by a distributed algorithm derived via formulating and solving the Lagrange dual problem of $\widetilde{\mathbf{P}}$. In order to achieve this, we first take a look at the Lagrangian of $\widetilde{\mathbf{P}}$:

$$L(\tilde{x}, \tilde{r}, \mu^\alpha, \mu^\beta) = \sum_{f \in F^S} U_f(1 - \exp(-\tilde{x}_f)) - \sum_{l \in \mathcal{L}} \mu^\alpha_l(\tilde{x}_{f(l)} - \tilde{r}_l) - \sum_{f \in F^A} \mu^\beta_f(\sum_{f_c \in C(f)} \tilde{x}_{f_c} - \tilde{x}_f).$$

In $L(\tilde{x}, \tilde{r}, \mu^\alpha, \mu^\beta)$, $\mu^\alpha := (\mu^\alpha_l, l \in \mathcal{L})$ and $\mu^\beta := (\mu^\beta_f, f \in \mathcal{F})$ are vectors of Lagrangian multipliers, corresponding to the transformed capacity constraint (2.7) and the transformed aggregation constraint (2.8), respectively. They are also interpreted as the "shadow prices" of the constraints, which can be understood as the "costs" a flow will be charged if it violates the constraints.

[3]For more detailed explanation on $\mu^{\alpha *}$, please refer to Section 2.5.

[4]Slater's condition can be guaranteed by assuming there exist vectors $\tilde{x} \in \widetilde{X}$ and $\tilde{r} \in \widetilde{\Pi}$ which satisfy all the constraints, i.e., strictly feasible points exist.

Since it can be derived that

$$\begin{aligned}\sum_{f\in F^A}\mu_f^\beta(\sum_{f_c\in C(f)}\tilde{x}_{f_c}-\tilde{x}_f) &= \sum_{f\in F^A}\mu_f^\beta\sum_{f_c\in C(f)}\tilde{x}_{f_c}-\sum_{f\in F^A}\mu_f^\beta\tilde{x}_f\\ &= \sum_{f\in\mathcal{F}}\mu_{\pi(f)}^\beta\tilde{x}_f-\sum_{f\in F^A}\mu_f^\beta\tilde{x}_f\\ &= \sum_{f\in F^S}\mu_{\pi(f)}^\beta\tilde{x}_f+\sum_{f\in F^A}(\mu_{\pi(f)}^\beta-\mu_f^\beta)\tilde{x}_f\end{aligned}$$

and $\sum_{l\in\mathcal{L}}\mu_l^\alpha\tilde{x}_{f(l)}=\sum_{f\in\mathcal{F}}\mu_{l(f)}^\alpha\tilde{x}_f$, we reorganize the Lagrangian as follows:

$$\begin{aligned}L(\tilde{x},\tilde{r},\mu^\alpha,\mu^\beta) = &\sum_{f\in F^S}\Big[U_f(1-\exp(-\tilde{x}_f))-(\mu_{l(f)}^\alpha+\mu_{\pi(f)}^\beta)\tilde{x}_f\Big]\\ &+\sum_{f\in F^A}\Big[(-\mu_{l(f)}^\alpha-\mu_{\pi(f)}^\beta+\mu_f^\beta)\tilde{x}_f\Big]+\sum_{l\in\mathcal{L}}\mu_l^\alpha\tilde{r}_l.\end{aligned}$$

The dual of the primal problem $\widetilde{\mathbf{P}}$ is:

$$\widetilde{\mathbf{D}}:\ \min_{\mu^\alpha,\mu^\beta\geq 0}\quad D(\mu^\alpha,\mu^\beta),$$

where the dual objective function $D(\mu^\alpha,\mu^\beta)$ is given as

$$D(\mu^\alpha,\mu^\beta):=\max_{\tilde{x}\in\widetilde{X},\ \tilde{r}\in\widetilde{\Pi}}\quad L(\tilde{x},\tilde{r},\mu^\alpha,\mu^\beta)$$

In the dual objective function, the Lagrangian multipliers (shadow prices) μ^α and μ^β, serve as the dual variables. Furthermore, $D(\mu^\alpha,\mu^\beta)$ can be decomposed into two separate optimization problems: $D(\mu^\alpha,\mu^\beta)=D_1(\mu^\alpha,\mu^\beta)+D_2(\mu^\alpha)$. $D_1(\mu^\alpha,\mu^\beta)$ and $D_2(\mu^\alpha)$ are defined below:

$$\begin{aligned}D_1(\mu^\alpha,\mu^\beta):=\max_{\tilde{x}\in\widetilde{X}}\quad &\sum_{f\in F^S}\Big[U_f(1-\exp(-\tilde{x}_f))-(\mu_{l(f)}^\alpha+\mu_{\pi(f)}^\beta)\tilde{x}_f\Big]\\ &+\sum_{f\in F^A}\Big[(-\mu_{l(f)}^\alpha-\mu_{\pi(f)}^\beta+\mu_f^\beta)\tilde{x}_f\Big]\\ D_2(\mu^\alpha):=\max_{\tilde{r}\in\widetilde{\Pi}}\quad &\sum_{l\in\mathcal{L}}\mu_l^\alpha\tilde{r}_l\end{aligned}$$

Among them, $D_1(\mu^\alpha,\mu^\beta)$ denotes the *rate allocation problem*, while $D_2(\mu^\alpha)$ is the *scheduling*

problem. In particular, the rate allocation problem aims at finding the rate of each source node that maximizes the aggregate utilities of all sources, subject to the constraint that the system is stable under some scheduling policy, while the scheduling problem focuses on finding a scheduling policy that stabilizes the system, for any rate vector of sources picked by the rate allocation problem. In the rest of this section, we will first elaborate on these two problems separately, and then explain how to develop a cross-layer joint design of them.

2.5.2 Interpretation of the Prices

Before proceeding with the decoupled problems, we first provide detailed explanation on the aforementioned shadow prices μ^α and μ^β. Theoretically, these prices represent the "costs" a flow will be charged if it violates the constraints. In practice, they imply the congestion information that the network elements need to share with each other, so that the traffic rates on different links of the network can be adjusted appropriately.

Let us first take a look at μ^α, which corresponds to the transformed capacity constraint in $\widetilde{\mathbf{P}}$. When a flow f violates this constraint (i.e., $\tilde{x}_{f(l)} > \tilde{r}_l$), if f increases its rate for $d\tilde{x}_{f(l)}$, a cost of $\mu_l^\alpha d\tilde{x}_{f(l)}$ should be charged. Next, we give a practical interpretation of this cost in the context of the original problem $\mathbf{P}$. Since $\tilde{x}_{f(l)} = -\ln(1 - x_{f(l)})$, it can be derived that $d\tilde{x}_{f(l)} = \frac{1}{1-x_{f(l)}} dx_{f(l)}$. Therefore, the cost charged with respect to $x_{f(l)}$ is $\mu_l^\alpha d\tilde{x}_{f(l)} = \mu_l^\alpha \frac{1}{1-x_{f(l)}} dx_{f(l)}$.

In this chapter, we call $\mu_l^\alpha \frac{1}{1-\hat{x}_{f(l)}}$ the *link price* of flow f when it passes data at a rate of $x_{f(l)} = \hat{x}_{f(l)}$. With link price, when a flow f violates the network capacity constraint (2.3) in the original problem $\mathbf{P}$, i.e., $x_{f(l)} > r_l$, the total cost it needs to pay can be calculated as follows: $\int_{r_l}^{x_{f(l)}} \mu_l^\alpha \frac{1}{1-\hat{x}_{f(l)}} d\hat{x}_{f(l)} = \mu_l^\alpha (\tilde{x}_{f(l)} - \tilde{r}_l)$. As can be seen, it is quantitatively equal to the cost calculated in the context of $\widetilde{\mathbf{P}}$, which is $\mu_l^\alpha (\tilde{x}_{f(l)} - \tilde{r}_l)$.

On the other hand, μ^β corresponds to the transformed aggregation constraint in $\widetilde{\mathbf{P}}$. When the aggregate transformed rates of f's children flows are larger than f's transformed rate (i.e., $\sum_{f_c \in C(f)} \tilde{x}_{f_c} > \tilde{x}_f$), the total cost paid by them to f for its efforts of aggregating packets is $\mu_f^\beta (\sum_{f_c \in C(f)} \tilde{x}_{f_c} - \tilde{x}_f)$.

When at optimality, according to the Karush-Kuhn-Tucker conditions, only the prices corresponding to active constraints are positive, which implies the price of an uncongested link is zero.

2.5.3 The Rate Allocation Problem

The rate allocation problem can be further divided as follows:

$$D_1(\mu^\alpha, \mu^\beta) = \sum_{f \in F^S} \max_{\tilde{x}_f \in \widetilde{X}_f} \Phi(\tilde{x}_f) + \sum_{f \in F^A} \max_{\tilde{x}_f \in \widetilde{X}_f} \Psi(\tilde{x}_f)$$

where

$$\Phi(\tilde{x}_f) = U_f(1 - \exp(-\tilde{x}_f)) - (\mu^\alpha_{l(f)} + \mu^\beta_{\pi(f)})\tilde{x}_f$$
$$\Psi(\tilde{x}_f) = (-\mu^\alpha_{l(f)} - \mu^\beta_{\pi(f)} + \mu^\beta_f)\tilde{x}_f.$$

In other words, the rate allocation problem can be solved through separately solving the optimization problem of each source flow (i.e., $\max_{\tilde{x}_f \in \widetilde{X}_f} \Phi(\tilde{x}_f)$), and each aggregation flow (i.e., $\max_{\tilde{x}_f \in \widetilde{X}_f} \Psi(\tilde{x}_f)$). Recall that μ^α and μ^β are the costs a flow will be charged if it violates the constraints. $\Phi(\tilde{x}_f)$ and $\Psi(\tilde{x}_f)$ actually represent the "net benefit" of a flow.

Let us first study the optimization problem of each source flow $f \in F^S$. As previously discussed, $U_f(1 - \exp(-\tilde{x}_f))$ is strictly concave and twice continuously differentiable. Consequently, $\Phi(\tilde{x}_f)$ is strictly concave and smooth, and thus has a unique maximizer when $\frac{d\Phi(\tilde{x}_f)}{d\tilde{x}_f} = 0$. Thus, given a valid utility function, the optimal solution can be easily identified. For example, assume $U_f(.) = \ln(.)$, it follows that $\frac{d\Phi(\tilde{x}_f)}{d\tilde{x}_f} = \frac{\exp(-\tilde{x}_f)}{1-\exp(-\tilde{x}_f)} - (\mu^\alpha_{l(f)} + \mu^\beta_{\pi(f)}) = 0$ from where the maximizer can be solved as below:

$$\tilde{x}^*_f = -\ln\Big(\frac{\mu^\alpha_{l(f)} + \mu^\beta_{\pi(f)}}{\mu^\alpha_{l(f)} + \mu^\beta_{\pi(f)} + 1}\Big).$$

When taking into account the feasible range of $\tilde{x}_f$, which is $\widetilde{X}_f = [-\ln(1-m_f), -\ln(1-M_f)]$, the optimal value of $\tilde{x}_f$ given μ^α and μ^β should be

$$\tilde{x}_f(\mu^\alpha, \mu^\beta) = \arg\max_{\tilde{x}_f \in \widetilde{X}_f} \Phi(\tilde{x}_f) = \begin{cases} \tilde{x}^*_f & \text{if } -\ln(1-m_f) \le \tilde{x}^*_f \le -\ln(1-M_f) \\ -\ln(1-m_f) & \text{if } \tilde{x}^*_f < -\ln(1-m_f) \\ -\ln(1-M_f) & \text{if } \tilde{x}^*_f > -\ln(1-M_f) \end{cases} \tag{2.9}$$

On the other hand, for each aggregation flow $f \in F^A$, since $\frac{d\Psi(\tilde{x}_f)}{d\tilde{x}_f} = -\mu^\alpha_{l(f)} - \mu^\beta_{\pi(f)} + \mu^\beta_f$ is a

constant, it follows that given the feasible range $\widetilde{X}_f = [0, -\ln(1 - M_f)]$, together with μ^α and μ^β, the optimal value of $\tilde{x}_f$ can be calculated as below:

$$\tilde{x}_f(\mu^\alpha, \mu^\beta) = \arg \max_{\tilde{x}_f \in \widetilde{X}_f} \Psi(\tilde{x}_f) = \begin{cases} 0 & \text{if } \mu_f^\beta < \mu_{l(f)}^\alpha + \mu_{\pi(f)}^\beta \\ -\ln(1 - M_f) & \text{if } \mu_f^\beta > \mu_{l(f)}^\alpha + \mu_{\pi(f)}^\beta \\ \text{any value in } \widetilde{X}_f & \text{otherwise} \end{cases}. \tag{2.10}$$

As previously discussed, strong duality holds in $\widetilde{\mathbf{P}}$, and thus there is no duality gap. Thereby, the optimal dual variables (prices) μ^α and μ^β exist (Proposition 5.1.4 in [5]), denoted as $\mu^{\alpha*}$ and $\mu^{\beta*}$. If $\mu^{\alpha*} > 0$ and $\mu^{\beta*} > 0$ are dual optimal, then $\tilde{x}_f(\mu^{\alpha*}, \mu^{\beta*})$ is also primal optimal, given that $\tilde{x}_f$ is primal feasible (Proposition 5.1.5 in [5]). In other words, once the optimal prices $\mu^{\alpha*}$ and $\mu^{\beta*}$ are available, the optimal rate $\tilde{x}_f^*$ can be achieved. The role of μ^α and μ^β is two-fold. First, they serve as the pricing signal for a flow to adjust its rate. Second, they decouple the primal problem, i.e., the global utility optimization into individual rate optimization of each flow.

2.5.4 The Scheduling Problem

We now turn to the scheduling problem $D_2(\mu^\alpha)$. It is actually a NP-hard problem, since it is equivalent to the maximum weighted independent set problem over the conflict graph. Actually, the conflict graph depends on the underlying interference model. In this chapter, we consider node-exclusive interference model, i.e., links that share a common node cannot transmit or receive simultaneously. This model has been used in many existing works [12, 53, 54] on network utility maximization. With the node exclusive interference model, the scheduling problem can be reduced to the maximum weighted matching problem, which is polynomial-time solvable. However, the existing polynomial-time solution [64] requires centralized implementation. In [35], a simple distributed approximate algorithm is presented, which is at most a factor of 2 away from the maximum, and has a linear running time $O(|\mathcal{L}|)$. We utilize this algorithm to solve the scheduling problem $D_2(\mu^\alpha)$ in a distributed manner.

Actually, the rate control strategy proposed in this chapter is a general framework and thus can be extended to other interference models. For any interference model, as long as an appropriate

algorithm can be designed to solve the scheduling problem $D_2(\mu^\alpha)$, it can be integrated with our framework.

Additionally, in some applications of sensor networks, the duty-cycle of the sensor nodes further complicate the scheduling problem [75,79]. We will try to address this challenge in our future work.

2.5.5 Subgradient Algorithm

Now let us see how we can minimize the dual objective function $D(\mu^\alpha, \mu^\beta)$. Gradient-based methods are, in general, attractive approaches to carry out minimizations of this type. Unfortunately, in our case, $D(\mu^\alpha, \mu^\beta)$ is nondifferentiable, and therefore its gradient may not always exist. This is because in general, differentiability of the dual requires a unique primal optimizer, whereas in our case, the optimal values of $\tilde{x}_f$ ($f \in F^A$) can be non-unique. Furthermore, $D_2(\mu^\alpha)$ is a piecewise linear function and not differentiable. Therefore, we choose to use subgradient method to solve this problem.

The subgradient algorithm that we propose next is based on the subgradient method developed by N. Z. Shor (Chapter 2 in [72]). In our problem, although the dual gradient does not exist, subgradients do. Based on Proposition 6.1.1 of [5], we adjust μ^α and μ^β in the opposite direction to the subgradients:

$$\begin{aligned} \mu_l^\alpha(t+1) &= \left[\mu_l^\alpha(t) - h(t)\frac{\partial D(\mu^\alpha(t), \mu^\beta(t))}{\partial \mu_l^\alpha}\right]^+ \\ &= \left[\mu_l^\alpha(t) + h(t)(\tilde{x}_{f(l)}(\mu^\alpha(t), \mu^\beta(t)) - \tilde{r}_l(\mu^\alpha(t)))\right]^+ \end{aligned} \tag{2.11}$$

$$\begin{aligned} \mu_f^\beta(t+1) &= \left[\mu_f^\beta(t) - h(t)\frac{\partial D(\mu^\alpha(t), \mu^\beta(t))}{\partial \mu_f^\beta}\right]^+ \\ &= \left[\mu_f^\beta(t) + h(t)(\sum_{f_c \in C(f)} \tilde{x}_{f_c}(\mu^\alpha(t), \mu^\beta(t)) - \tilde{x}_f(\mu^\alpha(t), \mu^\beta(t)))\right]^+ \end{aligned} \tag{2.12}$$

In the above formulas, the $\tilde{x}_f(\mu^\alpha, \mu^\beta)$ and $\tilde{r}_l(\mu^\alpha)$ are the maximizers of $D_1(\mu^\alpha, \mu^\beta)$ and $D_2(\mu^\alpha)$, given μ^α and μ^β; $h(t)$ is a positive scalar stepsize (note that the unit of t is time slot, not subslot); '+' denotes the projection onto the set $\mathbf{R}_+$ of non-negative real numbers.

Equation (2.11) reflects the law of supply and demand. If the demand of a flow f for bandwidth $\tilde{x}_{f(l)}$ exceeds its supply $\tilde{r}_l$, the transformed capacity constraint is violated. Thus, the price μ_l^α is raised. Otherwise, μ_l^α is reduced. Similarly, in (2.12), if the children flows $f_c \in C(f)$ demand an aggregate rate higher than the rate of its parent flow f, the transformed aggregation constraint is violated. Thus, the price μ_f^β is raised. Otherwise, μ_f^β is reduced.

2.5.6 Convergence Analysis

In this subsection, we justify the convergence property of the subgradient algorithm. Subgradient may not be a direction of descent, but makes an angle less than 90 degrees with all descent directions. Using results on the convergence of the subgradient method [5, 72], we show that, for a constant stepsize h, the algorithm is guaranteed to converge to within a neighborhood of the optimal value. The reason why we choose a constant stepsize is that it is convenient for distributed implementation. Since the usual convergence criterion is not applicable for a subgradient algorithm[5], we are interested in the asymptotical convergence. Similar to [12], we define $\overline{\mu}^\alpha(T) := \frac{1}{T}\sum_{t=1}^{T}\mu^\alpha(t)$ and $\overline{\mu}^\beta(T) := \frac{1}{T}\sum_{t=1}^{T}\mu^\beta(t)$ as the average dual variables by time T, and let $\overline{\tilde{x}} := \frac{1}{T}\sum_{t=1}^{T}\tilde{x}(t)$ be the average primal variable by time T. The following theorems guarantee the statistical convergence of the subgradient method. The proofs are similar to [12], and are omitted due to the limit of space.

Theorem 4. Let $\mu^{\alpha*}$ and $\mu^{\beta*}$ be the optimal dual variables, then, for some $0 < B < \infty$, the following inequality holds

$$\limsup_{T\to\infty} D(\overline{\mu}^\alpha, \overline{\mu}^\beta) - D(\mu^{\alpha*}, \mu^{\beta*}) \le hB. \tag{2.13}$$

Theorem 5. Let $\tilde{x}^*$ be the optimal rate of $\widetilde{\mathbf{P}}$, then, for some $0 < B < \infty$, the following inequality holds

$$\liminf_{T\to\infty} \widetilde{\mathbf{P}}(\overline{\tilde{x}}) \ge \widetilde{\mathbf{P}}(\tilde{x}^*) - hB. \tag{2.14}$$

The above theorems imply that the time-average primal and dual variables obtained by the subgradient algorithm can be made arbitrarily close to the optimal values if we choose the stepsize h sufficiently small.

[5]This is because the dual cost usually will not monotonically approach the optimal value, but wander around it under the subgradient algorithm.

2.6 Distributed Implementation

In this section, we describe how the subgradient algorithm can be implemented in a real network in a distributed and scalable way. In our design, a source (aggregation) node needs to communicate only with its parent and children nodes. In detail, each node collects the transformed rate $\tilde{x}$ from its children, and updates the prices (μ^{α} and μ^{β}) based on Eqn. (2.11) and Eqn. (2.12). Then, it broadcasts updated prices to its children. Upon receiving the price information from its parent, each node calculates its transformed rate based on Eqn. (2.9) or Eqn. (2.10). Then, it forwards its updated rate to its parent. Moreover, the nodes solve the scheduling problem through the distributed algorithm as we discussed previously in Section 2.5.4, and decide who will have a chance to transmit in the next slot. Before convergence, each node transmits at a rate $\hat{x} = \min(x, r)$. At each subslot, it must conform to the data availability constraint.

In our algorithm, in each iteration, an independent set is picked as the solution of the scheduling problem. From a long-term perspective, the algorithm jumps among the extreme points (i.e., r^i) of the capacity region (recall that each extreme point corresponds to an independent set.), and never touches the inner area. As aforementioned, the optimal solution (i.e., r^*) is the convex combination of $|\mathcal{L}|$ extreme points (i.e., $r^* = \sum_{i=1}^{|\mathcal{L}|} \alpha_i r^i$), located on a facet of the capacity region's boundary. In reality, each α_i is actually the percentage of iterations that the algorithm picks the ith independent set, after the system converges.

2.7 Discussions

2.7.1 Validity of Data Availability Constraint

As mentioned in Section 2.3.3, the data availability constraint is not taken into account when we solve the optimization problem **P**. However, this will not cause any problem as long as the rate of each flow converges to a feasible point. In an aggregation node, it maintains a queue for each of its children, and one more queue for the available data. Suppose the packets in each queue are sorted by their timestamps, as shown in Fig. 2.1 and Fig. 2.2. Thus, the height of each queue is determined by the timestamp (i.e., T_t) of the packet on the top of this queue. If the aggregation node behaves strictly following the data availability constraint, the queue of the available data should have the

same height as the shortest child queue. Clearly, after the optimal solution which satisfies both the network capacity constraint and data aggregation constraint is attained, the height of each child queue as well as the queue storing the available data will not grow infinitely.

Furthermore, our solution is suboptimal, and thus does not utilize the network resource to the extreme. Therefore, there is no doubt that the proposed scheme in this chapter will not overflow any node in the aggregation tree.

2.7.2 Periodic Data Collection

Some sensing tasks require periodic data collection, namely, the intervals between the timestamps of consecutive packets are fixed. In this case, if we further assume synchronized data collection, i.e., all the sources start their collection at the same time, we can achieve the largest time-overlap of the packets, and thus maximize the rate of each source node. However, in practice, the time-offsets of the packets from different nodes may be time-varying, due to the dynamic join (leave) of sensor nodes, and the oscillation of the rates caused by the variation of the environment as well as the underlying MAC layer scheduling. In this scenario, the proposed algorithm can be considered as a good approximation, since the probabilistic rate model can capture the long-term expectation of time-offsets. For example, in the scenario shown in Fig. 2.1, node A and B both collect data in a periodic pattern, and their rates are $\frac{1}{2}$ and $\frac{1}{4}$. There are two possibilities for the time-offset of the packets from A and B, as shown in Fig. 2.1(b) and Fig. 2.1(c). The rate of aggregated packets (i.e., node C's sending rate) in these two cases are $\frac{1}{2}$ and $\frac{3}{4}$, respectively. If either case has the same chance to happen, the expected rate of the aggregation flow is $\frac{1}{2} \times (\frac{1}{2} + \frac{3}{4}) = \frac{5}{8}$. This exactly equals the lower bound of node C's rate derived by the data aggregation constraint $(1-(1-\frac{1}{2})(1-\frac{1}{4}) = \frac{5}{8})$.

Even if the time-offsets can be controlled, however, in this scenario it is extremely difficult to mathematically model the relationship between a parent flow and its children flows in a convex function. Suppose the data collection are all synchronized, what we can do is to provide some tricks which can improve the objective value after the algorithm converges. In detail, we check the source flows sharing the same parent. If their periods are co-prime to each other, there is nothing can be improved since the data aggregation constraint precisely models the aggregation of the source flows with coprime periods. If the periods of some flows share a greatest common divisor α (let

F^α be the set of them), we fix their rates as constants in $\mathbf{P}$, and use a virtual flow f_α to replace them in the data aggregation constraint. f_α is resulted from aggregating the flows in F^α when they are synchronized, and its rate is the constant $x_{f_\alpha} = \frac{1}{\alpha}(1 - \prod_{f \in F^\alpha}(1 - \alpha x_f))$. Subsequently, we restart the optimization of $\mathbf{P}$. Since x_{f_α} is lower than the rate derived by the data aggregation constraint, some network resources are saved, and thus the rates of other flows can be improved. As an example, suppose the rates of the flows from node A and B shown in Fig. 2.1 are $\frac{1}{4}$ and $\frac{1}{6}$, we have $x_{f_\alpha} = \frac{1}{3}$ according to above formula. Obviously, x_{f_α} is lower than the rate obtained based on Eqn (2.5), which is $\frac{3}{8}$. Thus, some bandwidth can be saved from the flow originated at node C and allocated to its neighboring flows. After reoptimization, the rates of these neighboring flows will be improved.

2.7.3 Lossy Link

Due to the unliable nature of wireless communication, packets may be lost during transmission. In our scheme, lost packets do not matter at all, since from the perspective of the receiver, lost packets look like "nonexistent packets", namely, the source nodes never collect data at those subslots. Furthermore, if the average reception probability of each link can be measured, the formulation of the problem can be easily redefined so as to take it into account. Retransmissions are not needed in our solution.

2.7.4 Energy Constraint

Energy scarcity is a major challenge for the design of sensor networks. Our approach can also be adapted to address this problem. The solution is to add an energy constraint to the problem formulation. As a result, the energy budget of each node on the aggregation tree will be considered when the algorithm allocates the resource of the network.

2.7.5 Time Synchronization

The problem of synchronization has been considered and addressed by the prior work on data aggregation in sensor networks [59], and we just borrow the existing solutions.

2.8 Performance Evaluation

In this section, we provide simulation results to complement the analysis in the previous sections. We consider a randomly generated aggregation tree shown in Fig. 2.4. On this tree, 10 source nodes (shaded nodes) collect and forward data to the sink S, through 7 aggregation nodes. In addition, the number on each node (edge) works as the index of this node (flow). First, we assume that all the links have a normalized capacity of 0.5, and all the source nodes use the same utility function $U(x) = \ln(x)$. Then, we apply our joint rate control and scheduling algorithm with a fixed stepsize $h = 1$ on this aggregation tree, and observe its performance.

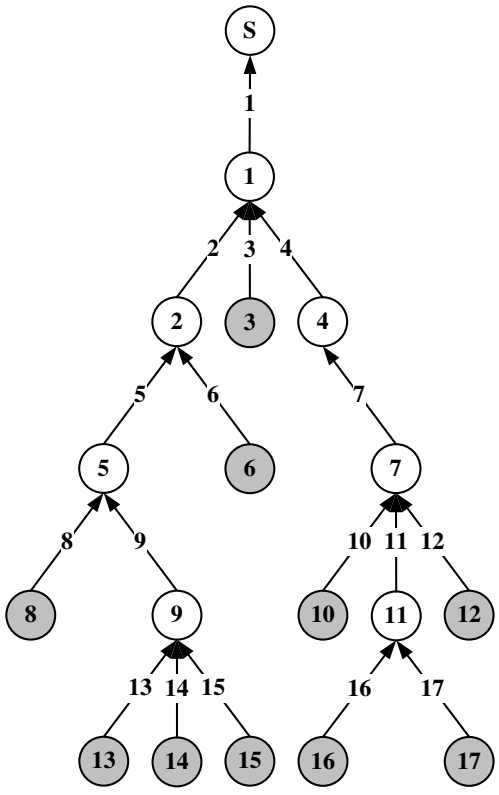
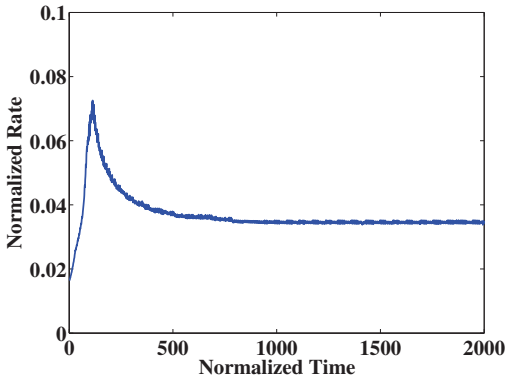
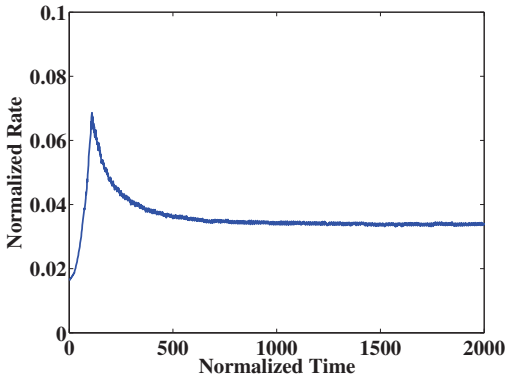


Figure 2.4: An aggregation tree



(a) Rate of flow 10

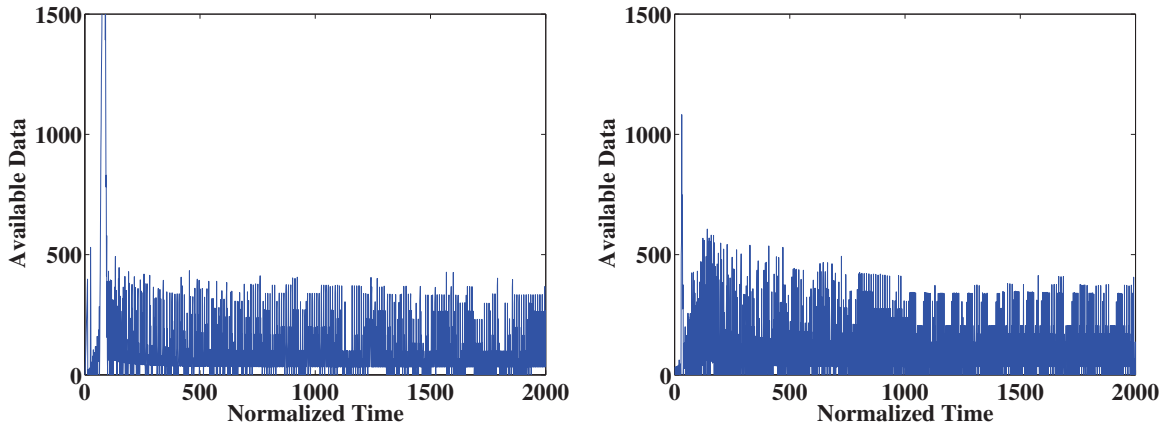


(b) Rate of flow 14

Figure 2.5: Rates of source flows

Figure 2.5 shows the evolution of the rates of the source flow 10 and 14. The other source flows have similar behavior and thus we omit their results. As one can see, they converge quickly to a neighborhood of the optimal values and oscillate around the optimal values. This oscillating behavior mathematically results from the nondifferentiability of the dual function and physically

can be interpreted as due to the scheduling process.



(a) Available data of node 5

(b) Available data of node 7

Figure 2.6: Available data stored in aggregation nodes

Figure 2.6 describes the amount of available data stored in node 5 and 7. The other nodes have similar behavior and thus we omit their results. In this test, we assume each time slot (length of step) contains 100 subslots. As can be seen, although the two curves both fluctuate with the time going, they are bounded reasonably. The rise of fluctuation can be ascribed to the underlying scheduling, which prevents an aggregation node from receiving and transmiting packets at the same time.

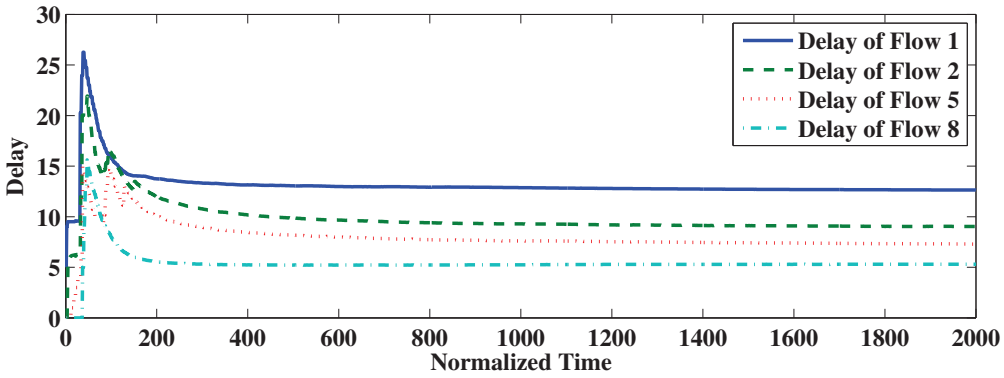
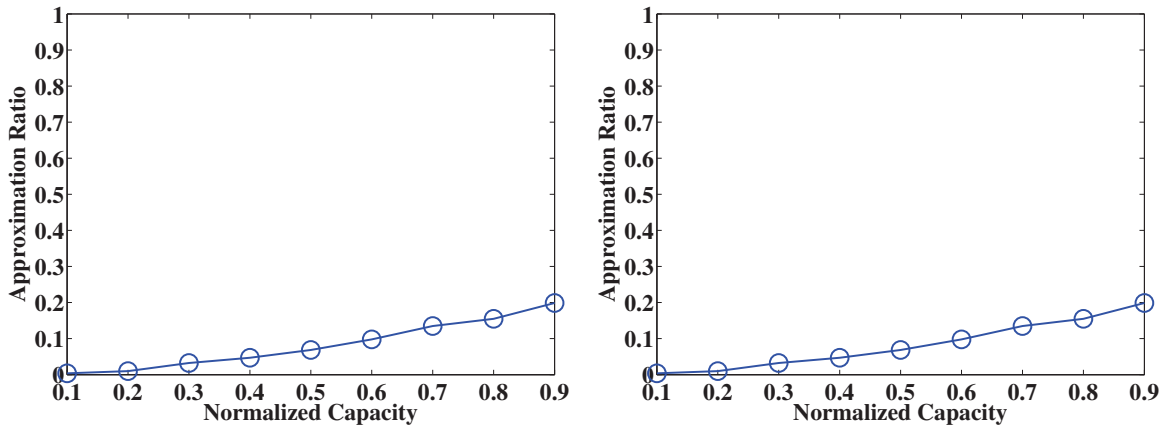


Figure 2.7: Delays of the packets delivered by flows

Figure 2.7 demonstrates the average delays of the packets delivered by four flows. Here the concept of delay is defined as the period from the moment when a packet is generated by a source to the time when this packet is delivered by a flow. For example, suppose node 8 generates a packet at time 1. Based on the delay values shown in Fig. 2.7, this packet will arrive at node 5 at time 6,

since the delay of flow 8 is roughly 5. Similarly, it will arrive at node 2 at time 8 since the delay of flow 5 is 7, and node 1 at time 10 since the delay of flow 2 is 9. Finally it will reach the sink at time 14 since the delay of flow 1 is 13. Here we should note that this packet may be aggregated during this process. As can be seen in the figure, the average delay of each flow converges to a stable point soon after the algorithm is started.



(a) Optimal Value

(b) Approximation Ratio

Figure 2.8: Optimal Solution V.S. Approximate Solution

Finally, Fig. 2.8 discloses the difference between the optimal objective value and the approximate objective value. In this test, we tune the capacity of each link, and observe its impact on the objective value. Intuitively, as illustrated in Fig. 2.8(a), both the optimal objective value and the approximate objective value increase with the capacities growing. At first, when the capacity of each link is as low as 0.1, the difference between the two values is negligible, since the approximation ratio defined as $\frac{|\text{Optimal}-\text{Approximate}|}{|\text{Approximate}|}$ is less than 1%, as shown in Fig. 2.8(b). As the capacities grow up, the difference as well as the approximation ratio increase accordingly. They remain in a low level (less than 10%) until the capacity goes beyond 0.5. Finally, the approximation ratio reaches around 20% when the capacity is increased to 0.9.

To find a reasonable explanation for this point, let us observe the function we use in all the variable substitutions, which is $f(x) = -\ln(1-x)$. The curvature, i.e., the second order derivative of this function is monotonously increasing with x. Thus, when the original variable x is small, the value of the new variable $y = f(x)$ is close to x. For this reason, when we decrease the capacity of each link, the boundary of the transformed region Π' on which the approximate solution (i.e., $\tilde{r}_0$

shown in Fig. 2.3(c)) is located will become closer to the boundary of the approximate region $\widetilde{\Pi}$ on which the optimal solution (i.e., $\tilde{r}^*$ shown in Fig. 2.3(c)) is located. Consequently, $\tilde{r}^* - \tilde{r}_0$ drops accordingly, resulting in a smaller difference between the optimal value and the approximate value, by Theorem 3. Finally, it should be noted that the optimal value shown in this test is actually the optimal value of the approximate problem (i.e., $\widetilde{\mathbf{P}}^*$), which has been proved to be the upper bound of the real optimal value $\mathbf{P}^*$. This implies that in practice, our approximate solution should be even closer to the real optimal solution than we observe in this experiment.

2.9 Related Work

In this section, we provide brief summaries of the existing work on sensory data aggregation and network utility maximization respectively, and clarify the novelty of this chapter.

Sensory data aggregation becomes a research hotspot after the presentation of the seminal work [59]. A large variety of problems regarding this topic have been extensively studied. Representative problems include: how to construct the most energy efficient aggregation tree [86, 92], how to schedule the transmissions of sensor nodes such that the aggregation delay can be minimized [87, 101], how to maximize or bound the lifetime of the aggregation tree [36, 94], how to secure data aggregation [66,98], how to derive theoretic bound of aggregation capacity [28,61], how to achieve fair aggregation [13, 22], etc.

The framework of network utility maximization (NUM) was first developed in the context of wireline network in [46,47], followed by [18,43,56]. The main idea of the framework is based on the decomposition of a system-wide optimization problem. A distributed congestion control mechanism is designed to drive the solutions of the decomposed problems towards the global optimum. Later on, NUM was studied in the context of wireless networks. In wireless networks, this problem is more difficult because of the interference nature of wireless communication. To address this challenge, a scheduling mechanism is integrated into the optimization framework to stabilize the system [12, 23, 53, 54].

This work is the first attempt to utilize NUM framework to explore the optimal rate allocation for sensory data aggregation. The theoretical and practical challenges aforementioned in Section 2.1 make the problem we target at completely different from previous work, and thus none of the

existing solutions can be applied.

2.10 Conclusions

In this chapter, we identify the unique challenges of rate allocation in the context of data aggregation in wireless sensor networks, and formulate this problem as a network utility maximization problem. After transforming this problem into a convex approximate problem, we decompose it based on the duality theory, and propose a distributed algorithm to solve the decoupled problems. Theoretical analysis and simulation results demonstrate the near-optimal performance of our scheme.

Chapter 3

Energy Efficient Decision Aggregation

3.1 Introduction

As we previously discussed, data aggregation can combine simple homogeneous raw data (e.g., temperature, humidity) from multiple sensor nodes to produce simple statistics such as average, max, and min. Now, we have two questions. First, is it possible to aggregate complex heterogeneous data such as audio and video clips? If the raw data are audio or video clips, it would be very expensive to transmit them. So the second the question is, can we further reduce the size of data sent by the sensor nodes, and thus save more resources?

The answer is yes! Many applications of cyber-physical systems or sensor networks are decision making applications, where a decision or choice need be made based on the information provided by the sensors. For example, we can use a set of acoustic sensors to identify the species of the observed birds based on their vocalizations. In the context of cyber-physical systems, species recognition like the above example is a typical category of decision making applications [9,21,38]. There are many other decision making applications. In target surveillance and tracking, sensor nodes should be able to identify different types of targets, such as cars, tanks, humans and animals [2,8,30]. In habitat monitoring, sensor nodes may distinguish different behaviors of monitored species [31, 60, 71]. In environmental monitoring, it may be desired that the environmental (e.g., weather) conditions be classified based on their impact on humans, animals, or crops [14]. In health care or assisted living, an intelligent sensing system may automatically evaluate the health status of residents, and react when they are in danger [55, 84, 102].

In decision making applications, there is no need for sensor nodes to send raw data. Instead, we can simply ask each sensor to report its local decision instead of raw data, and combine their decisions to improve decision accuracy. We call this process *decision aggregation*, as opposed to

previously discussed data aggregation.

To solve decision making problems, the most widely used technique may be classification [33,63, 80], which is the task of assigning objects (data) to one of several predefined categories (classes). Its basic idea is to use a function (also called classifier) "learned" from a set of training data in which each object has feature values and a class label to determine the class labels of newly arrived data. For example, consider the task of classifying bird species based on their vocalizations. Suppose we are interested in two bird species: Song sparrow and American crow, and consider two vocalization features: call intensity and call duration[1]. After learning from the training set, the following target function can be derived: 1) low call intensity & short call duration → Song sparrow, and 2) high call intensity & long call duration → American crow. Suppose there is a bird with unknown species, we can judge which class it belongs to by mapping its feature values to the corresponding class based on the learnt function.

However, applying classification techniques to solve decision making problems in cyber-physical systems is complicated by a unique challenge, which is insufficient labeled training data. The lack of labeled sensory data can be attributed to the remote, harsh, and sometimes even hostile locales where a sensor network is normally deployed as well as the continuous variation of the surveilled environment. In such scenarios, manually creating a large training set becomes extremely difficult. Without sufficiently large training set, the learned classifier may not be able to describe the characteristics of each class, and tends to make bad predictions on new data. In the area of data mining and machine learning, some techniques called semi-supervised classification have been proposed in order to deal with insufficient labels [33,63]. However, these schemes assume centralized storage and computation, and cannot be directly applied in the context of sensor networks where data are distributed over a large number of sensors.

One possible way to apply centralized classification techniques is to transport all the sensory data to the sink for offline analysis. However, it has been revealed that wireless transmission of a bit can require over 1000 times more energy than a single 32-bit computation [3]. Thus, in designing an energy-scarce sensing system, it is desired that each node locally process and reduce the raw data it collects as opposed to forwarding them directly to the sink [68,69], especially for some data-

[1]We use these two features simply as an illustration, in practice bird species classification requires much more complicated features as in the experiment presented in Section 3.6.

intensive applications such as audio or video based pattern recognition. This design philosophy has great challenges in traditional low-end sensing platforms such as Mica2 mote [34]. In these platforms, sensor nodes have limited processing power, memory, and energy and hence cannot support computation intensive algorithms. Recently, some powerful sensing systems [15, 29, 65, 97] are presented, making it feasible to place the classification task locally on individual nodes so that communication energy and interference can be significantly reduced.

To address the above challenges, we propose a *hierarchical aggregate classification* (HAC for short) protocol for cyber-physical systems, which is built on a hierarchical tree topology where all nodes detect the same events. In order to overcome the obstacles presented by insufficient labels, we suggest that sensor nodes conduct cluster analysis, which groups data points only based on the similarity of their feature values without any training. The clustering results can provide useful constraints for the task of classification when the labeled data is insufficient, since the data that have similar feature values are usually more likely to share the same class label. To reduce the amount of data delivered to the sink, we let each node locally cluster events and report only its clustering result (also called *decision* in this chapter) to the parent node instead of sending the raw data which are normally multi-dimensional numbers or audio/video files. The decisions are then aggregated along the tree through an efficient algorithm called *Decision-Aggregation* which can integrate the limited label information with the clustering results of multiple sensor nodes. Finally, the global consensus is reached at the sink node.

Additionally, to control the tradeoff between the communication energy and the classification accuracy, we design an extended version of HAC, called the *constrained hierarchical aggregate classification* (cHAC) protocol. cHAC can achieve more accurate classification results compared with HAC, at the cost of more energy consumption.

To demonstrate the advantages of our schemes, we conduct intensive experiments on not only synthetic data but also a real testbed. This testbed is a good example of high-end sensing system on which various data and computation intensive classification applications can be deployed. In our evaluation, we design two experiments on the testbed. The first one is to classify different bird species based on their vocalizations, and the second one is to predict the intensity of bird vocalizations as a function of different environmental parameters measured by the sensor nodes.

Both the experimental and simulation results show that the proposed protocols can achieve accurate classification in the face of insufficient label information, and provide a flexible option to tune the tradeoff between energy and accuracy.

The rest of the chapter is organized as follows. Section 3.2 introduces the general aggregation model. In Section 3.3, the Decision-Aggregation algorithm, which is the aggregation function used by the HAC and cHAC protocols at each nonleaf node, is presented. We explain how the HAC protocol utilizes this algorithm to aggregate the decisions along the tree in Section 3.4. In Section 3.5, the cHAC protocol, together with the procedures it invokes, is presented. The proposed schemes are evaluated in Section 3.6. Then, we summarize the related work in Section 3.7, and conclude the chapter in Section 3.8.

3.2 Aggregation Model

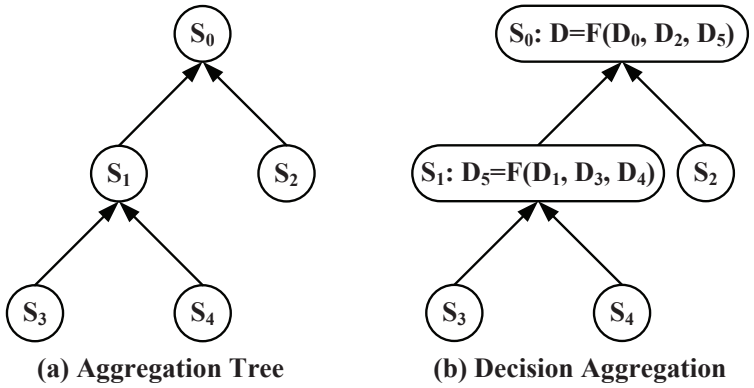


Figure 3.1: An example of decision aggregation

We consider an aggregation tree [48,59] T rooted at the sink node (or base station), and denote the set of sensor nodes on this tree by $\mathcal{S}_T = \{s_i,\ i = 1, 2, \ldots, n_T\}$. When an evpent takes place, all the nodes collect sensory readings about it[2]. Let $\mathcal{E} = \{e_i,\ i = 1, 2, ..., t\}$ denote the sequence of events (sorted in chronological order) detected by the sensor nodes. Suppose only a small portion of the events are labeled, and our goal is to find out the labels of the rest events. The general idea of our solution is as follows. Based on its sensory readings, each node, say s_i, divides the events into different clusters through its local clustering algorithm. After that, s_i forwards the

[2]We assume the aggregation tree is constructed on a set of nodes which are deployed in proximity. Thus, they can detect the same events and have similar readings.

clustering result (referred to as s_i's *decision*, denoted by D_i) to its parent node. At each nonleaf node (including the sink node), the decisions of its children nodes, together with its own decision if it has one, are further aggregated. Figure 3.1 gives an illustrative example of decision aggregation. As can be seen, the nonleaf node s_1 aggregates the decisions of s_3 and s_4, together with its own decision. In this chapter, we use function $\mathbf{F}(\cdot)$ to represent the operation of decision aggregation. Then, s_1 forwards the aggregated decision D_5 to the sink node s_0. At s_0, D_5 is further combined with s_0 and s_2's decisions. Finally, the global decision D is obtained. In the next section, we elaborate on the decision aggregation operation $\mathbf{F}(\cdot)$. Afterwards, we will disclose how the HAC and cHAC protocols invoke $\mathbf{F}(\cdot)$ to aggregate the decisions along the aggregation tree in Section 3.4 and Section 3.5, respectively.

3.3 Decision Aggregation

The decision aggregation function $\mathbf{F}(\cdot)$ takes the clustering decisions of multiple sensors as well as the label information as the input, and outputs a class label for each event, indicating which class the event belongs to. Although the clustering decisions do not give the concrete label assignment, they provide useful information for the classification task. $\mathbf{F}(\cdot)$ utilizes the labels from the few labeled events to guide the aggregation of clustering decisions such that a consolidated classification solution can be finally outputted. In this section, we first propose to model the decision aggregation problem as an optimization program over a bipartite graph. Then we present an effective solution and give performance analysis.

3.3.1 Belief Graph

Given a nonleaf node, suppose it receives n decisions from its children nodes. In each decision, the events in $\mathcal{E}$ are partitioned into m clusters[3]. Thus, we have totally $l = mn$ different clusters, denoted by $c_j,\ j = 1, 2, ..., l$. In this chapter, we call these clusters the *input clusters* (iCluster for short) of a decision aggregation. On the other hand, the decision aggregation operation will output an aggregated decision also composed of m clusters, named as *output clusters* (oCluster for short).

[3]Most of the clustering models like K-means can control the number of clusters. We let the number of clusters equal the number of classes of the events, which is m.

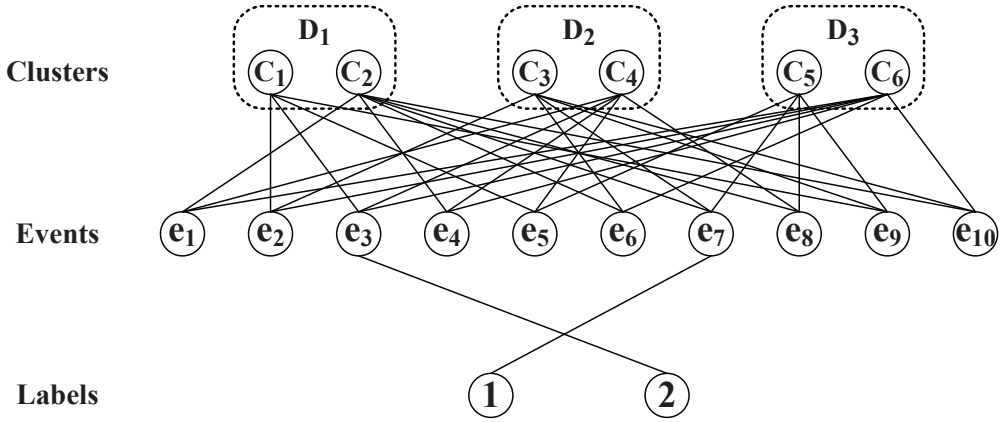


Figure 3.2: Belief graph of decision aggregation

In this chapter, we represent the relationship between the events and the iClusters as a bipartite graph, which is referred to as *belief graph*. In belief graph, each iCluster links to the events it contains. Figure 3.2 demonstrates the belief graph of a decision aggregation. In this case, we suppose there are $t = 10$ events, which belong to $m = 2$ different classes. Each of the sensor nodes partitions these 10 events into $m = 2$ clusters based on its local clustering algorithm, and there are $n = 3$ decisions. Thus, we have totally $l = mn = 6$ different iClusters. Moreover, to integrate label information into the belief graph, we add one more set of vertices, which represent the labels of the events. In belief graph, the labeled events are connected to the corresponding label vertices. As shown in Fig. 3.2, event e_3 and e_7 are labeled, and thus link with label vertex 2 and 1, respectively.

3.3.2 Terminology

The belief graph can be summarized by a couple of affinity matrices:

- Clustering matrix $A = (a_{ij})_{t\times l}$, which links events and iClusters as follows:

$$a_{ij} = \begin{cases} 1 & \text{If } e_i \text{ is assigned to cluster } c_j. \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

- Groundtruth matrix $Z_{t\times m} = (\vec{z}_{1\cdot}, \vec{z}_{2\cdot}, \ldots, \vec{z}_{t\cdot})^T$, which relates events to the label information.

For a labeled event e_i, its groundtruth vector is defined as:

$$z_{ik} = \begin{cases} 1 & \text{If } e_i\text{'s observed label is } k. \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

For each of the events without labels, we assign a zero vector $\vec{z}_{i\cdot} = \vec{0}$ to it.

Then, we define two sets of probability vectors that will work as the variables in the optimization problem formulated later in the next subsection:

- For an event e_i, Let L_i^e $(i = 1, 2, ..., t)$ denote the ID of the oCluster to which e_i is assigned, namely, the label of e_i. In our optimization problem, we aim at estimating the probability of e_i belonging to the k-th oCluster $(k = 1, 2, ..., m)$, i.e., $\hat{P}(L_i^e = k|e_i)$. Thus, each event is associated with a m-dimensional probability vector:

$$\vec{x}_{i\cdot} = \{(x_{ik})|x_{ik} = \hat{P}(L_i^e = k|e_i),\ k = 1, 2, ..., m\} \tag{3.3}$$

  Putting all the vectors together, we get a probability matrix $X_{t\times m} = (\vec{x}_{1\cdot}, \vec{x}_{2\cdot}, \ldots, \vec{x}_{t\cdot})^T$. After X is computed, we classify the i-th event into the k-th class if x_{ik} attains the maximum in $\vec{x}_{i\cdot}$.

- For an iCluster c_j, we also define a m-dimensional probability vector :

$$\vec{y}_{j\cdot} = \{(y_{jk})|y_{jk} = \hat{P}(L_j^c = k|c_j),\ k = 1, 2, ..., m\} \tag{3.4}$$

  where L_j^c is the ID of an oCluster. In practice, $\hat{P}(L_j^c = k|c_j)$ implies the probability that the majority of the events contained in c_j are assigned to the k-th oCluster. In theory, it will serve as an auxiliary variable in the optimization problem. Correspondingly, the probability matrix for all the iClusters is $Y_{l\times m} = (\vec{y}_{1\cdot}, \vec{y}_{2\cdot}, \ldots, \vec{y}_{l\cdot})^T$.

In our design, there is a weight parameter associated with each decision. Initially, each decision is manually assigned a weight based on the properties (e.g., sensing capability, residual energy, location, etc) of the sensor node who makes this decision. The weight of each node represents the

importance of the this node, and the nodes which can provide more accurate readings are assigned higher weights. The aggregated decision has a weight equal to the summation of the weights of input decisions. All the clusters within a decision have the same weight as the decision. In the rest of this chapter, we use w_j to denote the weight of cluster c_j[4]. Finally, let $b_i = \sum_{k=1}^{m} z_{ik}$ be a flag variable indicating whether e_i is labeled or not.

3.3.3 Problem Formulation

With the notations defined previously, we now formulate the decision aggregation problem as the following optimization program:

$$\mathbf{P}: \min_{X,Y} \quad \sum_{i=1}^{t}\sum_{j=1}^{l} a_{ij} w_j ||\vec{x}_{i\cdot} - \vec{y}_{j\cdot}||^2 + \alpha \sum_{i=1}^{t} b_i ||\vec{x}_{i\cdot} - \vec{z}_{i\cdot}||^2 \tag{3.5}$$

$$\text{s.t.} \quad \vec{x}_{i\cdot} \geq \vec{0}, \quad |\vec{x}_{i\cdot}| = 1 \quad \text{for} \quad i = 1, 2, ..., t \tag{3.6}$$

$$\vec{y}_{j\cdot} \geq \vec{0}, \quad |\vec{y}_{j\cdot}| = 1 \quad \text{for} \quad j = 1, 2, ..., l \tag{3.7}$$

where $||.||$ and $|.|$ denote a vector's L2 and L1 norm respectively, and α is a predefined parameter. To achieve consensus among multiple clustering decisions, we aim at finding the optimal probability vectors of the event nodes ($\vec{x}_{i\cdot}$) and the cluster nodes ($\vec{y}_{j\cdot}$) that can minimize the disagreement over the belief graph, and in the meanwhile, comply with the label information. Specifically, the first term in the objective function (Eqn. (3.5)) ensures that an event has similar probability vector as the input cluster to which it belongs, namely, $\vec{x}_{i\cdot}$ should be close to $\vec{y}_{j\cdot}$ if event e_i is connected to cluster c_j in the belief graph (e.g., event e_3 and cluster c_1, c_4, c_6 in Fig. 3.2). The second term puts the constraint that a labeled event's probability vector $\vec{x}_{i\cdot}$ should not deviate much from the corresponding groundtruth vector $\vec{z}_{i\cdot}$ (e.g., event e_3 and $\vec{z}_{3\cdot}$). α can be considered as the shadow price payment for violating this constraint. Additionally, since $\vec{x}_{i\cdot}$ and $\vec{y}_{j\cdot}$ are probability vectors, each of their components must be greater than or equal to 0 and the sum should equal 1.

[4]Sometimes, we use w_i to denote the weight of decision D_i or node s_i, and hope this causes no confusion.

3.3.4 Solution

By checking the quadratic coefficient matrix of the objective function, we can show that **P** is a convex program, which makes it possible to find a global optimal solution. We propose to solve **P** using the block coordinate descent method [5]. The basic idea of our solution is: At the τ-th iteration, we fix the values of $\vec{x}_{i\cdot}$ or $\vec{y}_{j\cdot}$, then the objective function of **P** becomes a convex function with respect to $\vec{y}_{j\cdot}$ or $\vec{x}_{i\cdot}$. Therefore, its minimum with respect to $\vec{x}_{i\cdot}$ and $\vec{y}_{j\cdot}$ can be obtained by setting the corresponding partial derivatives (i.e., $\frac{\partial f(X,Y)}{\partial x_{ik}}$ and $\frac{\partial f(X,Y)}{\partial y_{jk}}$, $k = 1, 2, \ldots, m$) to 0:

$$\vec{x}_{i\cdot}^{(\tau+1)} = \frac{\sum_{j=1}^{l} a_{ij} w_j \vec{y}_{j\cdot}^{(\tau)} + \alpha b_i \vec{z}_{i\cdot}}{\sum_{j=1}^{l} a_{ij} w_j + \alpha b_i}, \quad \vec{y}_{j\cdot}^{(\tau+1)} = \frac{\sum_{i=1}^{t} a_{ij} \vec{x}_{i\cdot}^{(\tau+1)}}{\sum_{i=1}^{t} a_{ij}} \tag{3.8}$$

The detailed steps are shown in Algorithm 1. The algorithm starts by initializing the probability matrix of input clusters randomly. The iterative process begins in line 3. First, the events receive the information (i.e., $\vec{y}_{j\cdot}$) from neighboring clusters and update $\vec{x}_{i\cdot}$ (line 5). Then, the events propagate the information (i.e., $\vec{x}_{i\cdot}$) to its neighboring clusters to update $\vec{y}_{j\cdot}$ (line 7). Finally, an event, say e_i, is assigned to the k-th oCluster if x_{ik} is the largest probability in $\vec{x}_{i\cdot}$ (line 10). According to [5] (Proposition 2.7.1), by showing the continuous differentiability of the objective function and the uniqueness of the minimum when fixing $\vec{x}_{i\cdot}$ or $\vec{y}_{j\cdot}$, we can prove that $(X^{(\tau)}, Y^{(\tau)})$ converges to the optimal point. When solving **P**, we don't take into account the constraints (Eqn. (3.6) and Eqn. (3.7)). By inductively checking the L1 norm of $\vec{x}_{i\cdot}^{(\tau)}$ and $\vec{y}_{j\cdot}^{(\tau)}$ from $\tau = 1$, it can be found out that the solution obtained by Algorithm 1 automatically satisfies the constraints.

Table 3.1 shows the first two iterations of the Decision-Aggregation algorithm (with $\alpha = 20$ and $w_j = 1$ for all j) for the belief graph shown in Fig. 3.2. We start with uniform probability vectors for the six clusters ($Y^{(1)}$). Then the probabilities of the events without labels are calculated by averaging the probabilities of the clusters they link to. At this step, they all have (0.5, 0.5) as their probability vectors. On the other hand, if the event is labeled (e.g., e_3 and e_7 in this example), the labeled information is incorporated into the probability vector computation where we average the probability vectors of the clusters each event links to and that of the groundtruth label (note that the vote from the true label has a weight α). For example, e_3 is adjacent to c_1, c_4, c_6 and label 2, so we have $\vec{x}_{3\cdot}^{(1)} = \frac{(0.5,0.5)+(0.5,0.5)+(0.5,0.5)+\alpha\cdot(0,1)}{3+\alpha} = (0.0652, 0.9348)$. During the second iteration,

Algorithm 1 Decision Aggregation

Input: Clustering matrix A, Groundtruth matrix Z, parameters α, set of weights $\mathcal{W}$, and ϵ;
Output: The class label for each event L_i^e;

Initialize $Y^{(0)}$, $Y^{(1)}$ randomly.
$\tau \leftarrow 1$
while $\sqrt{\sum_{j=1}^{l} ||\vec{y}_{j\cdot}^{(\tau)} - \vec{y}_{j\cdot}^{(\tau-1)}||^2} > \epsilon$ **do**
  for $i \leftarrow 1$ to t **do**
    $\vec{x}_{i\cdot}^{(\tau+1)} = \frac{\sum_{j=1}^{l} a_{ij} w_j \vec{y}_{j\cdot}^{(\tau)} + \alpha b_i \vec{z}_{i\cdot}}{\sum_{j=1}^{l} a_{ij} w_j + \alpha b_i}$
  end for
  for $j \leftarrow 1$ to l **do**
    $\vec{y}_{j\cdot}^{(\tau+1)} = \frac{\sum_{i=1}^{t} a_{ij} \vec{x}_{i\cdot}^{(\tau+1)}}{\sum_{i=1}^{t} a_{ij}}$
  end for
  $\tau \leftarrow \tau + 1$
end while
for $i \leftarrow 1$ to t **do**
  return $L_i^e \leftarrow \arg\max_k x_{ik}^{(\tau)}$
end for

$\vec{y}_{j\cdot}^{(2)}$ is obtained by averaging the probabilities of the events it contains. For example, $\vec{y}_{1\cdot}^{(2)}$ is the average of $\vec{x}_{2\cdot}^{(2)}$, $\vec{x}_{3\cdot}^{(2)}$, $\vec{x}_{5\cdot}^{(2)}$ and $\vec{y}_{9\cdot}^{(2)}$, which leads to $(0.3913, 0.6087)$. The propagation continues until convergence.

3.3.5 Performance Analysis

It can be seen that at each iteration, the algorithm takes $O(tlm) = O(tnm^2)$ time to compute the probability vectors of clusters and events. Also, the convergence rate of coordinate descent method is usually linear [5] (in practice, we fix the iteration number as a constant). Thus, the computational complexity of Algorithm 1 is actually linear with respect to the number of events (i.e., t), considering that the number of nodes involved in each aggregation (i.e., n) and the number of classes (i.e., m) are usually small. Thus, the proposed algorithm is not more expensive than the classification/clustering schemes, and thus can be applied to any platform running classification tasks. Furthermore, since wireless communication is the dominating factor of the energy consumption in sensor networks, our algorithm actually saves much more energy than it consumes.

Table 3.1: Iterations of Decision Aggregation

$Y^{(1)}$	$X^{(1)}$	$Y^{(2)}$	$X^{(2)}$
(0.5,0.5)	(0.5,0.5)	(0.3913,0.6087)	(0.4710,0.5290)
(0.5,0.5)	(0.5,0.5) (0.0652,0.9348)	(0.5725,0.4275)	(0.4686,0.5314) (0.0536,0.9464)
(0.5,0.5)	(0.5,0.5) (0.5,0.5)	(0.5870,0.4130)	(0.4710,0.5290) (0.4710,0.5290)
(0.5,0.5)	(0.5,0.5) (0.9348,0.0652)	(0.4130,0.5870)	(0.5290,0.4710) (0.9464,0.0536)
(0.5,0.5)	(0.5,0.5) (0.5,0.5)	(0.6087,0.3913)	(0.5314,0.4686) (0.5290,0.4710)
(0.5,0.5)	(0.5,0.5)	(0.4275,0.5725)	(0.5290,0.4710)

3.4 Hierarchical Aggregate Classification

Here we introduce the *Hierarchical Aggregate Classification* (HAC) protocol. HAC applies the Decision-Aggregation algorithm on each of the nonleaf nodes to aggregate all the decisions it collects. The output of the algorithm, i.e., the aggregated decision is forwarded upwards by the nonleaf node, and serves as one of the input decisions in the aggregation at its parent node. The message carrying the decision consists of t entries, corresponding to t events. In each entry, the index of an event and the ID of the oCluster to which this event belongs are stored. As previously defined, the ID of each oCluster is the label of this oCluster. However, these labels may not be useful in later aggregations, because the oClusters will probably be combined with other clusters whose labels are unknown. For instance, at the sink s_0 shown in Fig. 3.1(b), the labeled clusters in decision D_5 are combined with unlabeled clusters in D_0 and D_2. Finally, the global decision is obtained at the sink node, and each event is assigned a predefined label.

3.5 Constrained Hierarchical Aggregate Classification

In this section, we introduce an extended version of the HAC protocol, called the *Constrained Hierarchical Aggregate Classification* (cHAC) protocol. cHAC also uses the Decision-Aggregation algorithm as the aggregation function. However, different from HAC which requires that each nonleaf node aggregates all the decisions it collects, cHAC intelligently plans the decision aggregations

throughout the tree such that more accurate classification results can be obtained at the cost of more energy consumption. In the rest of this section, we first use a simple example to illustrate how energy and accuracy are correlated during the process of decision aggregation. Then we present the cHAC protocol, together with the procedures it invokes. Finally, the performance of cHAC is analyzed.

3.5.1 Tradeoff between Energy and Accuracy

Hierarchical aggregate classification, as discussed in the previous sections, can improve the classification accuracy as well as the consumption of communication energy through combining decisions coming from different sensor nodes. However, decision information is inevitably lost during the process of aggregation, and this may hurt the accuracy of the aggregated decision.

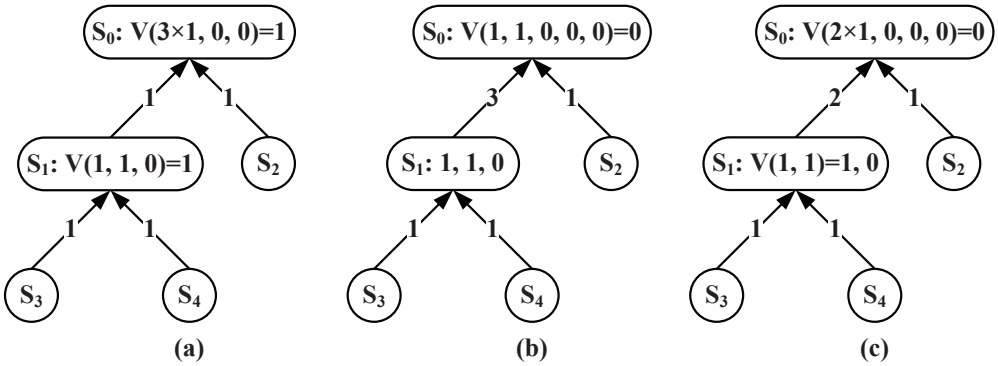


Figure 3.3: An example of energy-accuracy tradeoff

Let's continue to study the example shown in Fig. 3.1. Suppose all of the five sensors detect an event and try to predict the label of this event. To simplify the explanation, in this case we assume the sensor nodes are doing classification (not clustering). There are two classes with label 0 and 1 respectively. The local decisions of the nodes are: $D_0 = D_1 = D_2 = 0$ and $D_3 = D_4 = 1$ (recall that D_i is s_i's decision). Here we use a simple method, i.e., majority voting, as the aggregation function (denoted by $\mathbf{V}(\cdot)$). Note that only in this example we use majority voting as the aggregation function, in every other part of this chapter, the aggregation function means the Decision-Aggregation algorithm. Intuitively, given that the weight of each node is 1, the aggregated result of all the decisions should be 0, since there are more 0s than 1s among the 5 atomic decisions.

Figure. 3.3(a) illustrates the process of hierarchical aggregate classification along the aggregation

tree, where the numbers on the links imply the number of decisions transmitted along this link. At node s_1, D_1, D_3 and D_4 are combined. The resultant decision, which is D_5, is evaluated to be 1, since the majority of input decisions (D_3 and D_4) choose label 1. Then, D_5 is sent to the sink s_0, where the final aggregation happens. Since D_5 is the combination of three atomic decisions, its weight is the aggregate weight of three nodes, i.e., $w_5 = 3$. Therefore, the final aggregation is calculated as follows: $\mathbf{V}(w_5 D_5, w_0 D_0, w_2 D_2) = \mathbf{V}(3 \times 1, 0, 0) = 1$. Clearly, this result is not accurate, since more than half of the nodes predict the label of the event to be 0. The problem lies in the aggregation at node s_1, where some decision information, i.e., $D_1 = 0$ is lost[5].

3.5.2 Problem Formulation

To address this problem, we propose to trade energy for accuracy. Specifically, we add a constraint to the hierarchical aggregate classification, namely, in each decision aggregation along the tree (including the aggregation at the sink), the weight of each input decision cannot be larger than a predefined percentage of the total weight of all this aggregation's input decisions. Formally, suppose $\mathcal{D}$ is the set of the input decisions involved in an aggregation (note that $\mathcal{D}$ is NOT the set of all nodes' decisions), then it must be satisfied that $\frac{w_i}{\sum_{D_k \in \mathcal{D}} w_k} \leq \delta$ for $\forall\ D_i \in \mathcal{D}$, where w_i (w_k) denotes the weight of decision D_i (D_k), and δ is a predefined percentage. In this chapter, we call δ the *weight balance threshold*, and the constraint defined above the *weight balance constraint*. The weight balance threshold δ is a control parameter which can tune the tradeoff between energy and accuracy.

Intuitively, the smaller δ is, the larger number of decisions are combined in each aggregation, and thus the aggregated decision is closer to the global consensus. For example, if all the sensor nodes have the same weight and $\delta = \frac{1}{n}$ (n is the total number of sensor nodes), the weight balance constraint requires that each combination takes at least n decisions, which indicates that all the decisions need to be sent to the sink node without any aggregation on the half way. Clearly, the result of this aggregation perfectly represents the global consensus. Moreover, when δ is small, to

[5]Note that in this simple example, s_1 can send the numbers of 1s and 0s picked by its children (together with itself) to achieve better consensus compared with sending the majority voting result only. However, this solution cannot work for the decision aggregation problem where only clustering results are available. Since the same cluster ID may represent different classes in different decisions, we cannot simply count the number of labels assigned by the decisions. Furthermore, when the number of classes is large, this solution will consume excessive energy.

guarantee that a large number of decisions are combined in each aggregation, some decisions have to be transmitted for more than one hop along the aggregation tree, resulting in more transmissions.

For the simplicity of presentation, we assume that in each transmission, only one decision is forwarded. We are interested in the problem of *Constrained Hierarchical Aggregate Classification*: Under the weight balance constraint, among all the legal ways (solutions) which can aggregate the atomic decisions along the aggregation tree to reach a consensus at the sink, we want to pick the one with the minimum total number of transmissions. In fact, the hierarchical aggregate classification problem discussed in previous sections is equivalent to the case when $\delta = 1$.

Let's get back to the example shown in Fig. 3.3. Suppose in this case, δ is set to be 0.5. Apparently, the aforementioned solution (Fig. 3.3(a)) does not satisfy this constraint, since the weight percentage of D_5 in the final aggregation is more than half. Thus, although the absolute minimum transmission number (which is 4) is achieved in this case, it is not a valid solution. In contrast, a feasible solution is shown in Fig. 3.3(b). In this scenario, node s_1 does not make any aggregation, but directly forwards the decisions (D_1=0, D_3=1 and D_4=1) to the sink. This will surely satisfy the balance constraint. In addition, this solution actually achieves the highest accuracy, since no information is lost before arriving at the sink node. However, it results in unnecessary energy consumption (6 transmissions in total). Finally, Fig. 3.3(c) shows the optimal solution. Specifically, node s_1 combines two out of the three decisions ($D_5 = \mathbf{V}(D_3, D_4) = \mathbf{V}(1, 1) = 1$), and delivers the resultant decisions (D_1 and D_5) to the sink through 2 transmissions. This solution spends 5 transmissions, the minimum energy consumption that can be achieved under the weight balance constraint. More importantly, the global decision achieved by this solution is 0, which correctly represents the majority of the nodes.

As a matter of fact, the constrained hierarchical aggregate classification (cHAC) problem is an NP-complete problem. We prove this proposition by the following theorem:

Theorem 6. The constrained hierarchical aggregate classification problem is NP-complete.

Proof. First of all, we restate the cHAC problem as a decision problem. That is, we wish to determine whether the decision aggregation along a given tree can be done at the cost of exactly k transmissions. In this proof, we will show that the equal-size partition problem (ePartition for short), a known NP-complete problem, can be reduced to cHAC, i.e., ePartition $\leq_{\mathrm{P}}$ cHAC. The

equal-size partition problem is to decide whether a given set of integers can be partitioned into two "halves" that have both the same size (number of integers) and the same sum (summation of the integers).

The reduction algorithm begins with an instance of ePartition. Let $\mathcal{A} = \{a_1, a_2, \cdots, a_n\}$ $(n \geq 8)$ be a set of integers. We shall construct an instance of cHAC (denoted by Φ) such that $\mathcal{A}$ can be equally partitioned if and only if the answer to Φ is yes.

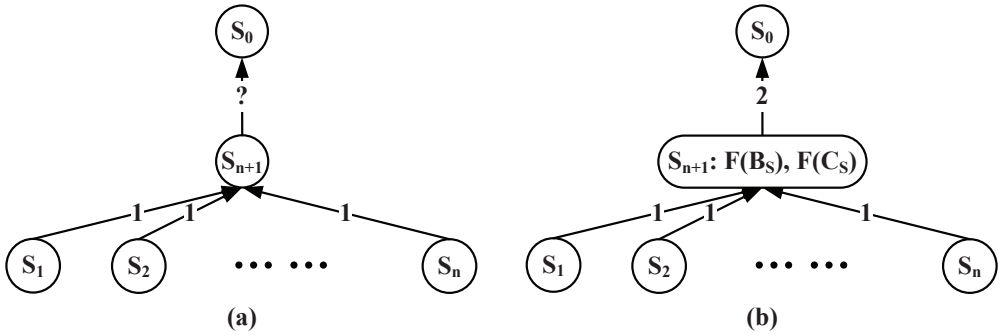


Figure 3.4: NP-completeness of cHAC

Φ is constructed as follows. An aggregation tree is shown in Fig. 3.4(a). The root node s_0 has a single child, which is s_{n+1}. There are n nodes $(s_1, s_2, \cdots, s_n)$ connected to s_{n+1}. Suppose the weight of node $s_i,\ i = 1, 2, \cdots, n$ is $a_i + M$, where M is a very large positive integer. Moreover, the weights of s_0 and s_{n+1} are $\frac{\sum_{i=1}^{n} a_i + nM}{2}$ and 0, respectively. In this case, the weight balance threshold is set to be $\delta = \frac{1}{3}$. Then, we introduce the formal description of Φ: Is there a way to solve the cHAC problem on the tree shown in Fig. 3.4(a) such that the total number of transmissions is exactly $n + 2$.

Suppose $\mathcal{A}$ can be partitioned into two equal-size subsets with equal summation. We denote these two subsets by $\mathcal{B}$ and $\mathcal{C}$. Without loss of generality, we suppose $\mathcal{B} = \{a_1, a_2, \cdots, a_{\frac{n}{2}}\}$ and $\mathcal{C} = \{a_{\frac{n}{2}+1}, a_{\frac{n}{2}+2}, \cdots, a_n\}$, and thus we have $\sum_{i=1}^{\frac{n}{2}} a_i = \sum_{j=\frac{n}{2}+1}^{n} a_j$. Correspondingly, we put nodes $s_1, s_2, \cdots, s_{\frac{n}{2}}$ in $\mathcal{B}_\mathcal{S}$ and $s_{\frac{n}{2}+1}, s_{\frac{n}{2}+2}, \cdots, s_n$ in $\mathcal{C}_\mathcal{S}$ (as shown in Fig. 3.4(b)). Since $w_i = a_i + M$, it can be derived that $\sum_{i=1}^{\frac{n}{2}} w_i = \sum_{j=\frac{n}{2}+1}^{n} w_j$, namely, $\mathcal{B}_\mathcal{S}$ and $\mathcal{C}_\mathcal{S}$ have the same total weight. In addition, no elements in $\mathcal{B}_\mathcal{S}$ and $\mathcal{C}_\mathcal{S}$ violate the weight balance constraint given that M is a large integer. Then, we combine decisions in $\mathcal{B}_\mathcal{S}$ and $\mathcal{C}_\mathcal{S}$ at node s_{n+1}, and use 2 transmissions to send the combined ones to s_{n+1}. Furthermore, since the weight of each combined decision is

$\frac{\sum_{i=1}^{n} a_i + nM}{2}$, which equals s_0's weight, the three decision (two combined decisions and s_0's decision) can be aggregated at node s_0 without violating the weight balance constraint (recall that $\delta = \frac{1}{3}$). Furthermore, since n transmissions are needed to move the clustering decision of each leaf node to node s_{n+1}, altogether $n + 2$ transmissions are used during this process, and thus the answer to Φ is yes.

Conversely, suppose the answer to Φ is yes. Since n transmissions (from s_i to s_{n+1}) are inevitable, we have to use 2 transmissions to send decisions from s_{n+1} to s_0. It is easy to see that the only way to achieve this is to combine the decisions at s_{n+1} into $\mathcal{B}_\mathcal{S}$ and $\mathcal{C}_\mathcal{S}$ with the same weight $\frac{\sum_{i=1}^{n} a_i + nM}{2}$, and then send them to s_0. For M is large, $\mathcal{B}_\mathcal{S}$ and $\mathcal{C}_\mathcal{S}$ must have the same size, and thus the corresponding halves $\mathcal{B}$ and $\mathcal{C}$ in $\mathcal{A}$ also have the same sum $\frac{\sum_{i=1}^{n} a_i}{2}$ (and of course, the same size). So Φ is yes implies that the ePartition instance is yes.

In summary, ePartition $\leq_\mathrm{P}$ cHAC is proved, and thus cHAC is NP-hard. Furthermore, it is straightforward to show that cHAC$\in$ NP, and thus a conclusion can be drawn that cHAC is NP-complete. □

The NP-completeness of the constrained decision aggregation problem makes it hopeless to find the optimal solution in polynomial time. In the rest of this section, we'll introduce an approximate solution, which is proved to have a constant approximation ratio and a polynomial complexity.

3.5.3 Constrained Decision Aggregation

In this subsection, we introduce the key function of our solution, which is called *constrained decision aggregation*. The constrained decision aggregation procedure works at each nonleaf tree node, except the sink. It partitions the decisions gathered at this node into different sets and invokes Decision-Aggregation to combine the decision sets which respect the weight balance constraint.

Intuitively, to guarantee that the final decision aggregation at the sink node is legal, each decision arriving at the sink should have a weight smaller than or equal to $W = \lfloor \delta W_T \rfloor$ (where W_T denotes the total weight of all the sensor nodes on the aggregation tree). Therefore, at any nonleaf node except the sink, the summation of the weights of all the input decisions involved in any aggregation must not exceed W. This is an additional constraint called the *weight summation constraint* for each nonleaf node. Also, W is referred to as the *weight summation threshold*. From

the analysis above, it is easy to see that any solution to the constrained hierarchical aggregate classification problem satisfies this constraint.

Consequently, at each nonleaf node, before aggregate the decisions, we need to solve the following problem first: Consider a nonleaf node s_0 with n children nodes $s_i,\ i = 1, 2, ..., n$. The goal is to divide the decision set $\mathcal{D} = \{D_1, D_2, ..., D_n\}$ into the minimum number of subsets such that each multi-decision subset respects both the weight summation constraint and the weight balance constraint. Since node s_0 spends one transmission to forward each aggregated subset or single-decision subset to its parent, minimizing the subset number implies the minimization of transmission number. To solve this problem, we introduce Decision-Selection, an algorithm which can pick a valid subset of decisions as long as there exists one. Afterwards, we give a formal definition of the Constrained-Decision-Aggregation procedure which iteratively invokes Decision-Selection and Decision-Aggregation to select and combine the decisions.

Decision-Selection is a dynamic programming based approach. First of all, we define a couple of notations. (a) $V[i, w]$: $V[i, w] = 1$ if out of the first i decisions in $\mathcal{D}$, it is possible to find a subset in which the aggregate weight of all the decisions is exactly w, and $V[i, w] = 0$ otherwise. (b) $keep[i, w]$: $keep[i, w] = 1$ if decision D_i is picked in the subset whose total weight is w, and $keep[i, w] = 0$ otherwise. The initial settings of $V[i, w]$ are described as below:

$$V[0, w] = 0 \qquad \text{for } 0 \leq w \leq W \tag{3.9}$$

$$V[i, w] = -\infty \qquad \text{for } w < 0 \tag{3.10}$$

$$V[i, w_i] = 1 \qquad \text{for } 1 \leq i \leq n \tag{3.11}$$

In Decision-Selection, $V[i, w]$ is recursively calculated based on Eqn. (3.12) for $1 \leq i \leq n$ and $0 \leq w \leq W$.

$$V[i, w] = \max(V[i-1, w], V[i-1, w-w_i]) \tag{3.12}$$

Algorithm 2 describes the detailed steps of Decision-Selection. Given a particular decision D_i and a weight sum w, what we are concerned about is under which condition D_i could be selected to the output decision set (i.e., set $keep[i, w]$ to be 1), which can not be directly seen from Eqn. (3.12). There are two possible cases. Case 1 happens in line 8. In this case, among the first $i-1$ decisions,

Algorithm 2 Decision Selection

Input: Weight summation threshold W, set of weights $\mathcal{W}$, set of input decisions $\mathcal{D}$, weight balance threshold δ;
Output: Set of decisions $\mathcal{A}$ satisfying both weight summation constraint and weight balance constraint;

```
 1: A ← Φ
 2: for w ← 0 to W do
 3:     V[0, w] ← 0;
 4: end for
 5: for i ← 1 to n do
 6:     for w ← 0 to W do
 7:         if w_i < w and V[i − 1, w − w_i] > V[i − 1, w] then
 8:             V[i, w] ← V[i − 1, w − w_i];
 9:             keep[i, w] ← 1;
10:         else if w_i = w then
11:             V[i, w] ← 1;
12:             if V[i − 1, w] = 0 then
13:                 keep[i, w] ← 1;
14:             else
15:                 keep[i, w] ← 0;
16:             end if
17:         else
18:             V[i, w] ← V[i − 1, w];
19:             keep[i, w] ← 0;
20:         end if
21:     end for
22: end for
23: for w ← W downto 1 do
24:     m ← w;
25:     for i ← n downto 1 do
26:         if keep[i, m] = 1 and w_i ≤ ⌊δw⌋ then
27:             A ← A ⋃ {D_i};
28:             m ← m − w_i;
29:         end if
30:     end for
31:     if A ≠ Φ then
32:         break;
33:     end if
34: end for
35: return A
```

we can find a subset with total weight $w - w_i$ (i.e., $V[i-1, w-w_i] = 1$), but cannot find a subset with total weight w (i.e., $V[i-1, w] = 0$). Obviously, D_i should be selected; Case 2 is in line 12. In this case, though $w = w_i$, we put D_i into the selected set only when no subset among the first $i-1$ decisions has a total weight of w (i.e., $V[i-1, w] = 0$), since the algorithm only picks one set for a particular weight sum. Decision-Selection has an important prerequisite: $\mathcal{D}$ *must be sorted in the ascending order of weight.* The motivation of this prerequisite can be better understood via Theorem 7.

Theorem 7. Decision-Selection can return a valid decision set satisfying both the weight summa-

Algorithm 3 Constrained Decision Aggregation

Input: Set of input decisions $\mathcal{D}$
Output: Set of output decisions Ω

1: Sort $\mathcal{D}$ in the ascending order of weight;
2: **repeat**
3: $\mathcal{A} \leftarrow$ Decision-Selection($\mathcal{D}$);
4: $\mathcal{D} \leftarrow \mathcal{D} - \mathcal{A}$;
5: $\mathcal{C} \leftarrow \mathcal{C} \bigcup \{$Decision-Aggregation($\mathcal{A}$)$\}$;
6: **until** $|\mathcal{A}| = 0$
7: $\mathcal{R} \leftarrow \mathcal{D}$;
8: $\Omega \leftarrow \mathcal{C} \bigcup \mathcal{R}$;
9: **return** Ω

tion constraint and the weight balance constraint, as long as there exists such a set in $\mathcal{D}$.

Proof. There is no doubt that given a number $1 < w < W$, Algorithm 2 can identify a subset of $\mathcal{D}$ whose weight summation is exactly equal to w, if such a subset really exists. There are at most W subsets found by the algorithm (stored in $keep[i, w]$), and all of them satisfy the weight summation constraint. Thus, we only need to show that if none of these selected subsets can satisfy the weight balance constraint, there does not exist a legal decision subset in $\mathcal{D}$. The key point is, given a weight summation w, there may exist multiple valid subsets, and among them, the subset (denoted by $\mathcal{D}^*(w)$) whose last decision has the smallest weight is the most likely to satisfy the weight balance constraint. This is because the decisions are sorted in the ascending order of the weight. Thus, given a decision set, if the last decision, which has the largest weight, satisfies the constraint, all the preceding decisions in this set satisfy the constraint as well. The Decision-Selection algorithm guarantees to pick $\mathcal{D}^*(w)$, since a decision D_i is selected (i.e., $keep[i, w] \leftarrow 1$, line 8 and 12) only when no valid subset exists among the first $i - 1$ decisions (i.e., $V[i - 1, w] = 0$, line 6 and 11). For example, suppose we have a sorted decision set $\mathcal{D} = \{w_1 = 2, w_2 = 3, w_3 = 4, w_4 = 7\}$, with the weight sum $w = 9$ and the balance threshold $\delta = \frac{1}{2}$. It is obvious that there are two subsets of $\mathcal{D}$ whose weight sum equals w. They are $\mathcal{D}_1 = \{2, 3, 4\}$ and $\mathcal{D}_2 = \{2, 7\}$. Among them, only $\mathcal{D}_1$, whose last decision has a smaller weight, can satisfy the weight balance constraint. In the algorithm, $keep[3, 9]$ will be set by 1, and $keep[4, 9]$ is assigned to be 0 since $V[3, 9] = 1$. Finally, lines 18-25 exhaustively check $\mathcal{D}^*(w)$ (line 21) with w ranging from W down to 1, and thus will not miss a valid subset if it does exist. □

With Decision-Selection, we are able to design the Constrained-Decision-Aggregation algorithm. As shown in Algorithm 3, after sorting $\mathcal{D}$ in the ascending order of weight, Constrained-Decision-Aggregation iteratively invokes Decision-Selection (line 3), and combines the returned decision set (i.e., $\mathcal{A}$) through the procedure of Decision-Aggregation (line 5). The resultant aggregated decisions are stored in a set $\mathcal{C}$. This process repeats until no more set is found by Decision-Selection, then the residual decisions left in $\mathcal{D}$ are moved to another set $\mathcal{R}$. In line 8, the union of $\mathcal{C}$ and $\mathcal{R}$ forms the set of output decisions Ω. Finally, $|\Omega|$ transmissions are consumed by the subtree root s_0 to forward the output decisions in Ω to its parent node.

3.5.4 Constrained Hierarchical Aggregate Classification

The constrained hierarchical aggregate classification (cHAC) protocol invokes the Constrained-Decision-Aggregation procedure at each nonleaf node except the sink, and aggregates the decisions along the tree. Specifically, suppose n subtrees are connected to a nonleaf node s_0. cHAC applies Constrained-Decision-Aggregation to each of the subtrees T_i, resulting in a set of aggregated decisions $\mathcal{C}_i$ and a set of residual decisions $\mathcal{R}_i$. After arriving at s_0, these sets form two union sets, which are $\mathcal{C}_0 = \bigcup_{i=1}^{n} \mathcal{C}_i$ and $\mathcal{R}_0 = \bigcup_{i=1}^{n} \mathcal{R}_i$. Then, cHAC employs Constrained-Decision-Aggregation on $\mathcal{R}_0$, and puts the newly aggregated decisions in $\mathcal{C}_0$. After the algorithm terminates, no decisions left in $\mathcal{R}_0$ can be further combined. Subsequently, s_0 spends $|\Omega_0| = |\mathcal{C}_0| + |\mathcal{R}_0|$ transmissions to forward the decisions. Altogether, $\sum_{i=0}^{n} |\Omega_i|$ transmissions are consumed during this process. In each transmission, the cHAC protocol uses the same message format as the HAC protocol to carry the decision.

At the sink node, the procedure of Decision-Aggregation (not Constrained-Decision-Aggregation) is called, since none of the arrived decisions has a weight larger than $W = \lfloor \delta W_T \rfloor$. Finally, the global consensus is achieved. It is easy to see, the cHAC protocol can be easily implemented in a distributed manner. Each node only needs to collect the decision information from its children and make local aggregations. Therefore, no global coordination is needed.

3.5.5 Performance Analysis

In this subsection, we show the approximation ratio and the computational complexity of not only the Constrained-Decision-Aggregation algorithm but also the whole constrained hierarchical aggregate classification process. We start with the analysis of the Constrained-Decision-Aggregation algorithm. First of all, we have the following observation.

Lemma 1. After Constrained-Decision-Aggregation terminates, there are at most one decision in $\mathcal{C}$ whose weight is no more than $\frac{W}{2}$.

The basic idea of the proof is: suppose there are two such decisions, they are resulted from two decision subsets whose weight summation is no more than $\frac{W}{2}$. However, the Decision-Selection algorithm should have combined these two subsets into one which satisfies both the weight balance constraint and the weight summation constraint. In fact, if there exists a decision in $\mathcal{C}$ with a weight less than or equal to $\frac{W}{2}$, we move it from $\mathcal{C}$ to $\mathcal{R}$.

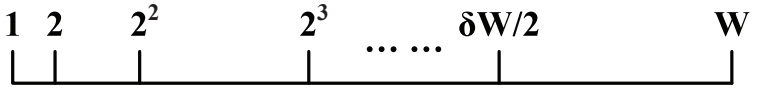


Figure 3.5: Intervals

According to Algorithm 3, the residual set $\mathcal{R}$ contains the decisions that cannot be aggregated. We project these decisions onto an interval $[1, W]$ based on their weights. Then, we divide the interval $[1, W]$ into subintervals by the points $2^i,\ i = 1, 2, ..., \lfloor \log_2(\delta W/2) \rfloor$ (together with the point $\delta W/2$), as shown in Fig. 3.5. Before proving the approximation ratio of Constrained-Decision-Aggregation in Theorem 8, we first prove the following claim in Lemma 2.

Lemma 2. Within each interval, there are less than $\frac{2}{\delta}$ decisions in $\mathcal{R}$.

Proof. By contradiction, suppose there are $\frac{2}{\delta}$ decisions within an interval delimited by the point 2^{i-1} and 2^i. The sum of their weights is less than W, for $i \leq \lfloor \log_2(\delta W/2) \rfloor$. Within $[2^{i-1}, 2^i]$, the aggregate weight of these decisions is at least $\frac{2}{\delta} \cdot 2^{i-1} = \frac{2^i}{\delta}$, while the weight of a single decision is at most 2^i. Thus, the weight percentage of any decision in this interval is at most δ, which satisfies the weight balance constraint. This contradicts with the fact that Constrained-Decision-Aggregation leaves them uncombined, so the proof is completed. □

Theorem 8. Suppose OPT is the transmission number in the optimal solution to the constrained decision aggregation problem, and SOL is the transmission number in the solution found by the Constrained-Decision-Aggregation algorithm. We have $\text{SOL} \leq \frac{2}{\delta} \cdot \big(\text{OPT} + \log_2 \frac{\delta W}{2}\big)$.

Proof. First of all, we define some notations. Let $W_{\mathcal{D}}$ denote the total weight of the decisions in $\mathcal{D}$. In addition, $\mathcal{R}_{\leq}$ and $\mathcal{R}_{>}$ are two subsets of $\mathcal{R}$ in which the weights of decisions are smaller than or equal to $\delta W/2$ and larger than $\delta W/2$, respectively. An lower bound of OPT is $W_{\mathcal{D}}/W$, since every (aggregated) decision has a weight of at most W. In our algorithm, decisions in $\mathcal{D}$ are partitioned and aggregated into two sets $\mathcal{C}$ and $\mathcal{R}$, and we have $\text{SOL} = |\mathcal{C}| + |\mathcal{R}| = |\mathcal{C}| + |\mathcal{R}_{\leq}| + |\mathcal{R}_{>}|$. For all the decisions in $\mathcal{C}$ whose weights are at least $W/2$ and all the decision in $\mathcal{R}_{>}$ whose weights are at least $\delta W/2$, we have $|\mathcal{C}| + |\mathcal{R}_{>}| \leq \frac{W_{\mathcal{D}}}{\delta W/2} = \frac{2}{\delta} \cdot \frac{W_{\mathcal{D}}}{W} \leq \frac{2}{\delta} \cdot \text{OPT}$, for $\text{OPT} \geq W_{\mathcal{D}}/W$. In addition, by Lemma 2, we have $|\mathcal{R}_{\leq}| \leq \frac{2}{\delta} \cdot \log_2 \frac{\delta W}{2}$. Thus, combining the above two inequalities, we can derive the promised bound $\text{SOL} \leq \frac{2}{\delta} \cdot \big(\text{OPT} + \log_2 \frac{\delta W}{2}\big)$. □

Then, the computational complexity of Constrained-Decision-Aggregation is given by Theorem 9.

Theorem 9. Constrained-Decision-Aggregation has a computational complexity of $O(n^2 \delta W)$[6].

Proof. First of all, Constrained-Decision-Aggregation sorts the decisions in $\mathcal{D}$, which takes a running time of $O(n \log n)$. Then, in the loop between line 2 and 6, Decision-Selection is repeatedly called, and each takes $O(nW)$. Since under the weight balance constraint, the number of decisions picked by Decision-Selection in each iteration must be no less than $\frac{1}{\delta}$, the number of times that Decision-Selection is called is at most δn. Therefore, the overall computational complexity of the Constrained-Decision-Aggregation procedure is $O(n \log(n)) + \delta n O(nW) = O(n^2 \delta W)$. □

Next, we give the approximation ratio and the computational complexity of the whole constrained hierarchical aggregate classification process by Theorem 10 and Theorem 11, respectively.

Theorem 10. Suppose OPT is the transmission number in the optimal solution to the constrained hierarchical aggregate classification problem, and SOL is the transmission number in the solution found by the cHAC protocol. Then, we have $\text{SOL} \leq \frac{2}{\delta} \cdot \big(1 + \log_2 \frac{\delta W}{2}\big) \cdot \text{OPT}$.

[6] In this analysis, we do not consider Decision-Aggregation, since it can be decoupled from Constrained-Decision-Aggregation.

Proof. The proof is similar to the proof of Theorem 8 in spirit, but the bound we derived is a bit weaker. Suppose the tree has n nodes (excluding the sink node). Let $\mathrm{OPT} = \mathrm{OPT}_1 + \mathrm{OPT}_2 + \ldots + \mathrm{OPT}_n$, where OPT_i is the number of transmissions from node s_i to its parent in the optimal solution. Similarly, $\mathrm{SOL} = \mathrm{SOL}_1 + \mathrm{SOL}_2 + \ldots + \mathrm{SOL}_n$, where SOL_i is the number of transmissions from s_i to its parent in the solution obtained by the cHAC protocol.

In case that s_i is a nonleaf node, the cHAC protocol takes the aggregated decisions from its children as the input. Intuitively, the lower bound of OPT_i is the optimal solution (denoted by $\widetilde{\mathrm{OPT}}_i$) to the problem that takes all the atomic decisions without being aggregated as the input. Thus, similar to the analysis in Theorem 8, we have $\mathrm{SOL}_i \leq \frac{2}{\delta} \cdot \left(\widetilde{\mathrm{OPT}}_i + \log_2 \frac{\delta W}{2}\right) \leq \frac{2}{\delta} \cdot \left(1 + \log_2 \frac{\delta W}{2}\right) \cdot \widetilde{\mathrm{OPT}}_i$. If s_i is a leaf node, it is apparent that $\mathrm{SOL}_i = \mathrm{OPT}_i$. Since $\widetilde{\mathrm{OPT}}_i \leq \mathrm{OPT}_i$, summing them up for $i = 1, 2, \ldots, n$, we can derive the promised bound $\mathrm{SOL} \leq \frac{2}{\delta} \cdot \left(1 + \log_2 \frac{\delta W}{2}\right) \cdot \widetilde{\mathrm{OPT}} \leq \frac{2}{\delta} \cdot \left(1 + \log_2 \frac{\delta W}{2}\right) \cdot \mathrm{OPT}$. □

Theorem 11. The cHAC protocol has a computational complexity of $O(n_T^2 W)$.

Recall that here n_T denotes the total number of nodes on the aggregation tree. In the worst case, the Decision-Selection algorithm is called for $O(n_T)$ times, and each takes $O(n_T W)$ time. Therefore, the overall computational complexity of the cHAC protocol is $O(n_T^2 W)$.

3.6 Performance Evaluation

In this section, we evaluate the proposed schemes on i) Synthetic data, and ii) A solar-powered sensor network testbed. For comparison, we design two baseline methods. Both of the baselines adopt the strategy of majority-voting, and they are different in the ways of generating votes. The first baseline method, which we call *Clustering Voting*, suggests that each node locally groups the events into different clusters (this part is the same as our scheme), and then count the labeled events (i.e., how many events belong to a particular label) in each cluster. Within a cluster, all the events are assigned the label with the largest count. For example, suppose there are totally 100 events in a cluster, with three events labeled 1 and two events labeled 0, then all the 100 events are labeled 1 according to the clustering voting scheme. Finally, the nodes vote to decide the label of each event. The second baseline, called *Classification Voting*, lets each node apply classification

algorithms (such as decision tree, SVM) on the labeled data, and predict the labels of the rest. Then, the events are labeled based on the vote. The detailed experimental results are shown and explained in the next two subsections.

3.6.1 Experiment on Synthetic Data

In this part, we evaluate our schemes on synthetic data. First of all, we randomly build aggregation trees with the number of tree nodes ranging from 20 to 80. The height of the trees increases with the augment of tree size. In particular, the trees of height 3, 4, 5 and 6 contain around 20, 30, 50 and 80 nodes, correspondingly. In order to achieve diversity, we apply different clustering algorithms (such as K-means, spectral clustering) to different nodes. Suppose each node has a weight between 0 and 5. For each height, we evaluate different tree topologies and record the average results.

Next, we give a brief description on how the synthetic data is generated. In this experiment, we assume there are 10 different types of sensors, corresponding to 10 features of the events (e.g., temperature, humidity, etc). Suppose the events are drawn from 5 different classes (labels), and we randomly assign the groundtruth labels (from 5 classes) to 10000 events. For each event, based on its assigned label, we generate its feature values from a Gaussian distribution in a 10-dimensional (each dimension corresponds to a feature) data space $\mathbb{R}^{10}$. Therefore, the collection of the events are drawn from a Gaussian mixture model with 5 components, each of which corresponds to a class. After creating the events, we generate the sensory readings of the nodes. For each node on the tree, we randomly assign a subset of the previously defined (10 types of) sensors to it. For each type of sensor assigned to this node, we generate its sensory reading of each event as follows: we first copy the corresponding feature value of the event and then add random Gaussian noise to this value. In this way, different nodes with the same types of sensors would have different sensory readings.

Figure 3.6 compares the classification accuracy (percentage of the correctly classified events) of the proposed hierarchical aggregate classification (HAC) and constrained hierarchical aggregate classification (cHAC) (with weight balance threshold $\delta = 1/2$, $1/3$, and $1/4$, respectively) protocols, and the two baseline schemes. As can be seen, when the percentage of labeled events is less than 10, the proposed protocols can always achieve better performance than the baselines. Moreover,

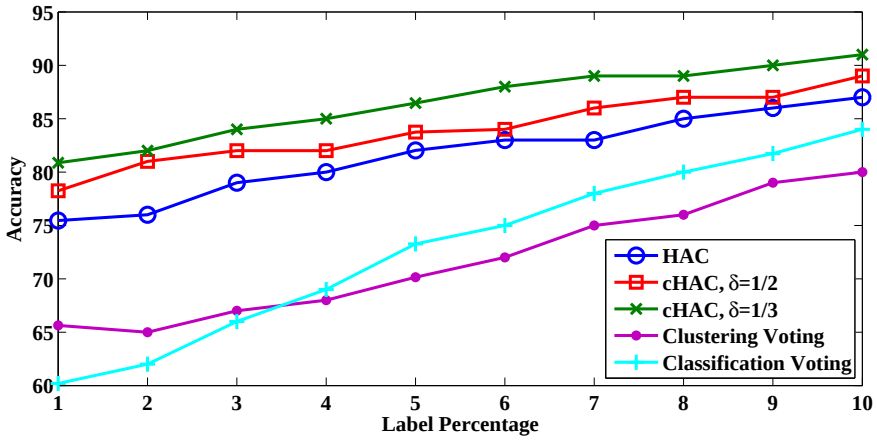
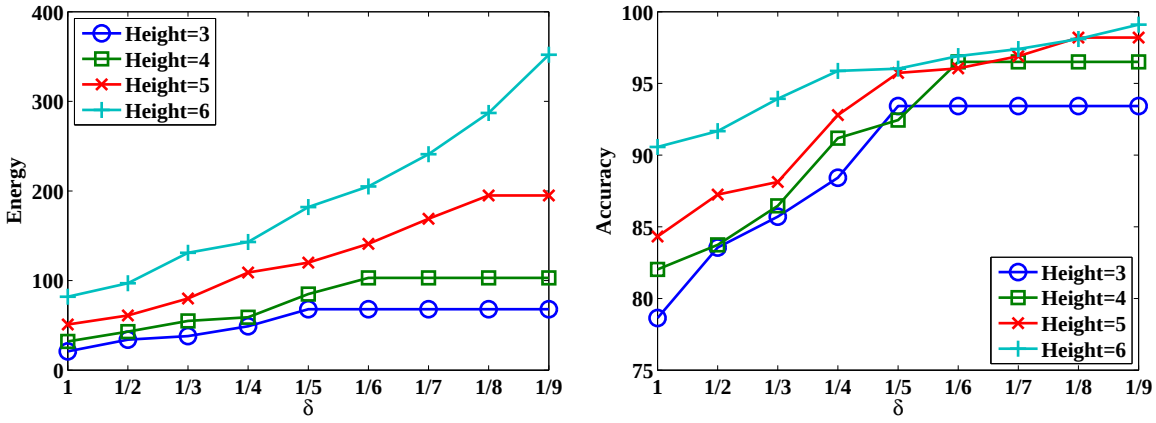


Figure 3.6: Comparison of accuracy (Synthetic data)

with the decrease of label percentage, the accuracy of all the five methods degrade. Among them, the accuracy of the classification voting decreases the fastest. This is reasonable since compared with clustering methods, classification models are normally more sensitive to the label percentage. Another notable point is that the cHAC with smaller δ can achieve higher accuracy.



(a) δ on energy

(b) δ on accuracy

Figure 3.7: Impact of δ on energy and accuracy (Synthetic data)

Figure 3.7 demonstrates the impact of weight balance threshold δ on the communication energy (in terms of the total number of messages transmitted by the tree nodes) and the classification accuracy of the cHAC protocol. In this experiment, we assume 5% of the events are labeled, and test four groups of aggregation trees, with height 3, 4, 5 and 6 respectively. As expected, when δ decreases, no matter of the tree size, more energy is consumed (Figure 3.7(a)), and higher accuracy can be achieved (Figure 3.7(b)). This confirms our scheme's capability of trading energy

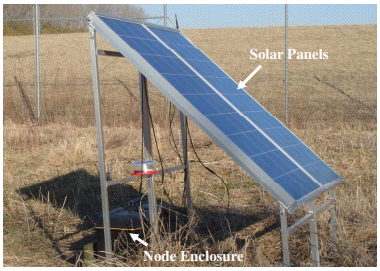


Figure 3.8: Outside look of a solar-powered sensor node

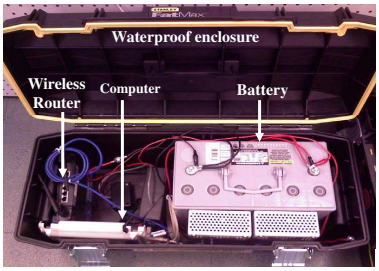


Figure 3.9: Inside look of a solar-powered sensor node

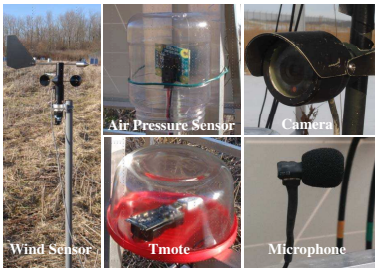


Figure 3.10: Different types of sensors on the nodes

for accuracy. Furthermore, given a δ, the trees with larger height tend to have higher accuracy, however, at the cost of more energy consumption. This is because usually better diversity can be obtained when more nodes are involved in the aggregation. Finally, when δ becomes lower than a threshold (e.g., $\delta = \frac{1}{5}$ for height-3 trees), the accuracy cannot be improved any more, since all the atomic decisions (locally made by each node) have been sent to the sink.

3.6.2 Experiment on Real Testbed

In this part, we test the performance of the proposed protocols on our solar-powered sensor network testbed [97]. This outdoor testbed is located on the edge of a forest. Currently, 9 nodes have been deployed and running since August 2008. Figure 3.8 and Figure 3.9 show the outside and inside look of a node, which comprises of a low-power PC to provide computing capability as well as a wireless router to support wireless communication among nodes. The nodes are equipped with multiple types of sensors, and thus can provide a broad spectrum of sensing capabilities for different environmental monitoring applications. Figure 3.10 shows some of the sensors integrated with the nodes, which can collect the sensory readings of temperature, humidity, light, wind speed, wind direction, and air pressure. In addition, the nodes are also equipped with microphones and cameras which are used to record audio and video information of wildlife. Readers can refer to [97] for more details on the system software and hardware architecture, as well as some implementation issues.

We construct an aggregation tree on the 9 deployed nodes, as shown in Fig. 3.11. In this tree, node 5 works as the sink, and all the nodes have weight 1. Furthermore, to avoid packet collision and overhearing during the process of decision aggregation, we employ a distributed aggregation scheduling algorithm proposed by [101]. Under this scheduling strategy, at any time slot only a

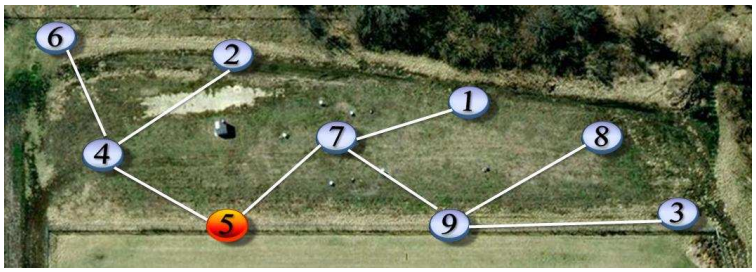


Figure 3.11: Tree topology of the 9 deployed nodes

subset of the sensor nodes are allowed to send packets and their transmissions do not interfere with each other. The wireless interfaces of the rest nodes are shut down so as to save the energy of idle listening.

To illustrate our approach, we design two experiments on this testbed, which are explained respectively later in this section.

Classification of Bird Species

In this experiment, we target on automatically recognizing different bird species based on their vocalizations. Bird species classification is a typical pattern recognition problem and has been extensively studied in recent years [1, 9, 24, 73]. Building bird species recognition system normally involves two phases: feature extraction phase and classification phase. In the feature extraction phase, bird vocalizations are represented with a few acoustical parameters (features) of the sound. Here the philosophy is that features should be selected so that they are able to maximally distinguish sounds produced by different bird species (classes). The most widely used parametrization method is the model of Mel-frequency cepstral coefficients (MFCC), which is adopted in this experiment. After the features are extracted, each audio data point is reduced to a vector of features. Subsequently, in the classification phase, classification or clustering algorithms can be directly applied on the feature vectors.

Three bird species: *song sparrow*, *American crow*, and *red-winged blackbird* are studied in this test. They are among the most frequently observed species around the place where the testbed is deployed. We select 4000 time periods within each of which the vocalizations of one species are recorded by all the sensor nodes of the testbed. The duration of each period is 1.5 seconds. The goal of this experiment is to determine the bird species (class) by which the vocalizations are

produced within each time period, given that only a small percentage of the 4000 time periods are labeled with the above three species.

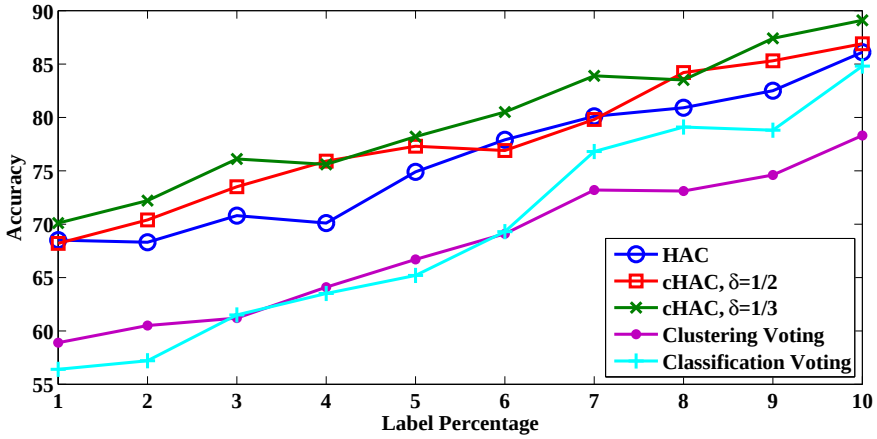
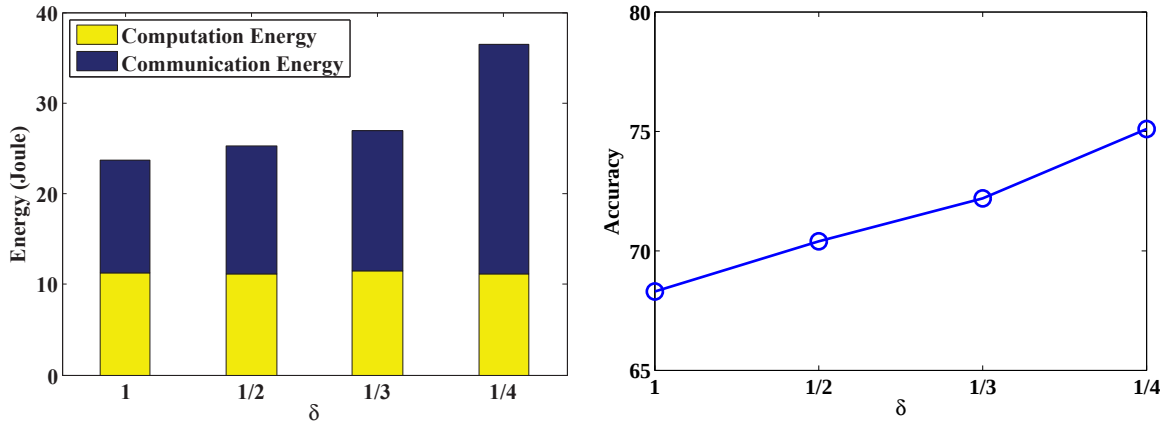


Figure 3.12: Comparison of accuracy (Species classification)

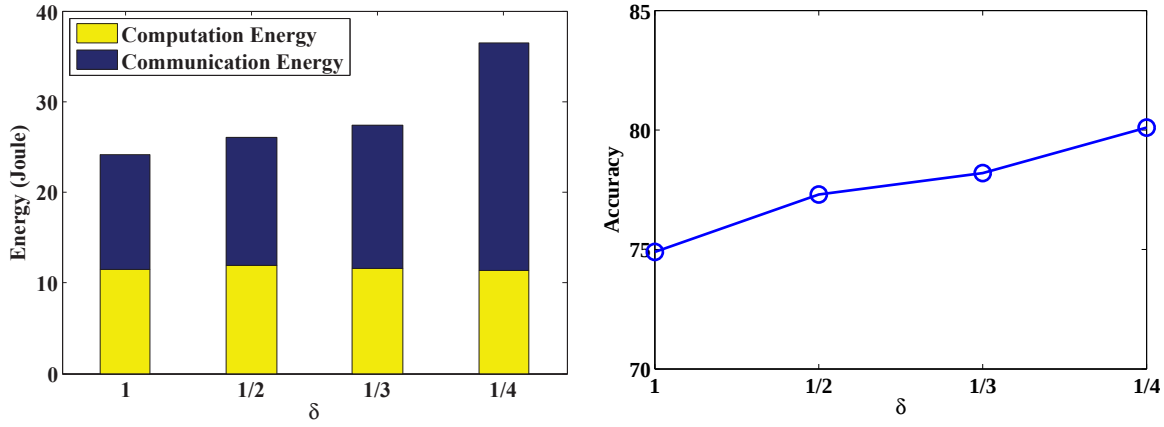
Initially, each sensor node locally extracts the MFCC features of each audio clip. Then we apply the same five methods (i.e., the proposed protocols and two baseline schemes) as in the preceding experiment to the extracted feature vectors, and observe their performance. The accuracies achieved by the five methods are shown in Fig. 3.12. Due to the unexpected noise in both the feature values and the labels, the curves are not as smooth as those shown in the experiment on synthetic data. However, we can still observe the same curve patterns as discovered in the previous experiment. In short, the proposed schemes always perform better than the baseline methods given a label percentage less than 10.

Different from the experiment on synthetic data in which energy consumption is approximated by the number of transmissions, here we measure the real energy consumption of not only communication but also computation on the testbed. In particular, the computation energy include the energy consumed by the classification (or clustering) algorithms as well as the HAC (or cHAC) protocol. We do not take into account the energy consumption of sensing since it is usually smaller than that of computation and communication, and more importantly, the sensing energy is the same no matter what kind of classification strategy is used. On the other hand, the communication energy is referred to as the energy consumption of sensor nodes by transmitting or receiving packets. As previously discussed, the extra energy expenditure caused by overhearing and idle listening is eliminated by carefully scheduling the on/off of each node's wireless interface.



(a) δ on energy (LP=2)

(b) δ on accuracy (LP=2)



(c) δ on energy (LP=5)

(d) δ on accuracy (LP=5)

Figure 3.13: Impact of δ on energy and accuracy (Species classification)

Figure 3.13 demonstrates the tradeoff between energy and accuracy tuned by the weight balance threshold δ (when the cHAC protocol is applied). In this test, we study two scenarios when the percentage of labeled data is 2 (LP=2) and 5 (LP=5), respectively. Since in this case the aggregation tree has only 9 nodes, there are only four possible choices on δ. However, the relation between energy and accuracy can still be clearly observed. Figure 3.13(a) and (c) describe the impact of δ on the total computation energy as well as the total communication energy under two label percentages. As one can see, in either case, the computation energy is nearly invariant regardless of δ, since the clustering algorithm locally executed by each node dominates this category of energy consumption. In contrast, the communication energy increases with the decrease of δ, resulting in the growth of total energy expenditure. On the other hand, as shown in Fig. 3.13(b) and (d), the classification accuracy is improved when δ goes down. Therefore, our expectation that the

cHAC protocol could trade energy for accuracy is realized. As a comparison, we also measure the energy needed to transport all the extracted feature vectors to the sink. The number is 1073.41 joules, which is significantly larger than the energy consumption of the proposed schemes. This is because the MFCC features are usually of high dimension and thus it costs enormous energy to deliver them. Therefore, it is difficult for the energy-scare sensing systems to afford centralized classification.

Classification of Bird Vocalization Intensity

We design the second experiment to test the proposed schemes on more types of sensors. We build a classifier that attempts to predict the intensity of bird vocalizations as a function of the aforementioned six environmental parameters (features): temperature, humidity, light, wind speed, wind direction and air pressure. The ground-truth intensity of bird vocalizations can be measured using microphones located on the nodes. In this experiment, we define three classes (labels), corresponding to three levels of vocalization intensity: 1) High intensity, 2) Medium intensity, and 3) Low intensity. The objective of this experiment is as follows. Given the collected environmental data, among which a small percentage has been labeled into the above three categories, we want to decide the labels of the remaining data. The experiment spans a period of one month. Every 10 minutes, the sensors of each node record the sensory readings corresponding to six features of the environment. Thus, at the end of the month, each node has collected about 4000 event readings. During this month, the intensity of bird vocalizations is also measured and averaged over 10 minute intervals. This average value is taken as the ground truth.

In this experiment, we test one more baseline scheme, called *Data Aggregation*. Different from the five schemes evaluated in the preceding experiments, instead of aggregating the decisions, data aggregation directly transports and averages the raw data along the aggregation tree, and applies centralized classification techniques on the averaged data at the sink. The comparison results of accuracy are exhibited in Fig. 3.14. As can be seen, the accuracy of data aggregation is higher than that of the HAC protocol, but lower than the accuracy of cHAC when all the decisions are delivered to the sink (i.e., $\delta = 1/4$). This is because some information is lost during the process of data aggregation. More importantly, data aggregation consumes about 64.87 joules of energy to

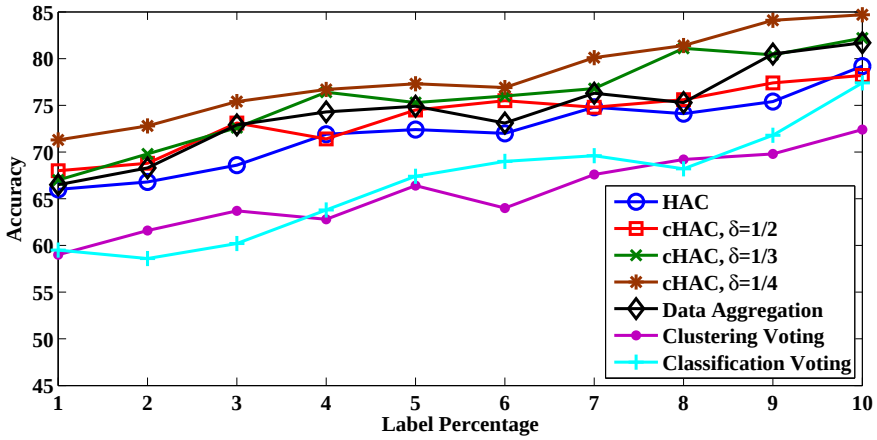


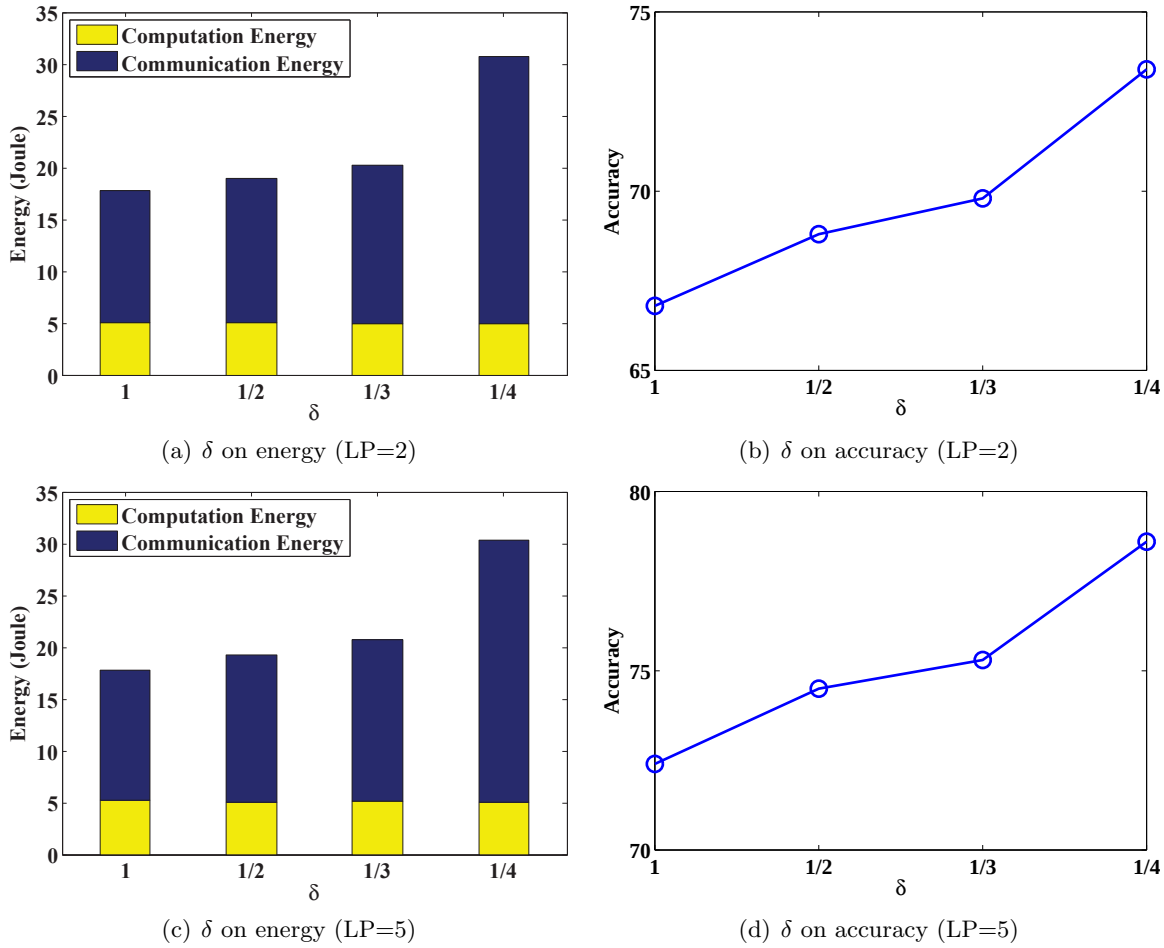
Figure 3.14: Comparison of accuracy (Intensity classification)

deliver the sensory readings to the sink, much larger than the communication energy consumed by the proposed protocols. The reason is that for each event, along each tree link our solution forwards only the index of the class to which the event belongs, while data aggregation transmits a vector of up to six numbers. Thus, data aggregation is not an economic solution from the perspective of energy.

The tradeoff between energy and accuracy can be observed in Fig. 3.15. In this experiment, the ratio of communication energy over computation energy is larger than that of the species classification experiment. This is because in this case the dimension of data (which is 6) is smaller, and thus the clustering algorithms consume less energy. Clearly, by tuning weight balance threshold δ, the cHAC protocol can trade energy for accuracy.

3.7 Related Work

In sensor networks, data reduction strategies aim at reducing the amount of data sent by each node [69]. Traditional data reduction techniques [16, 68, 69] select a subset of sensory readings that is delivered to the sink such that the original observation data can be reconstructed within some user-defined accuracy. For example, [69] presents a data reduction strategy that exploits the Least-Mean-Square (LMS) to predict sensory readings without prior knowledge or statistical modeling of the sensory readings. The prediction is made at both the sensor nodes and the sink, and each node only needs to send the readings that deviate from the prediction. [16] explores the



(a) δ on energy (LP=2)

(b) δ on accuracy (LP=2)

(c) δ on energy (LP=5)

(d) δ on accuracy (LP=5)

Figure 3.15: Impact of δ on energy and accuracy (Intensity classification)

use of a centralized predictive filtering algorithm to reduce the amount of transmitted data. It eliminates the predictor on sensor nodes. Instead, it relies on a low level signaling system at the sink that instructs the nodes to transmit their data when required. In contrast, the proposed decision aggregation algorithms summarize the sensory readings by grouping similar events into the same clusters, and report only the clustering results to the sink. The proposed schemes focus on the similarity among the data, and hence can be regarded as decision level data reduction.

As discussed in the introduction, classification techniques for sensor networks have been widely studied. For example, [14] studies hierarchical data classification in sensor networks. In this chapter, local classifiers built by individual sensors are iteratively enhanced along the routing path, by strategically combining generated pseudo data and new local data. However, similar to other classification schemes for sensor networks, this chapter assumes that a large amount

of labeled data are available, which is impractical in sensor networks. To solve the problem of learning from distributed data sets, people have designed methods that can learn multiple classifiers from local data sets in a distributed environment and then combine local classifiers into a global model [11,50,57]. However, these methods still focus on learning from a training set with sufficient labeled examples.

Moreover, the proposed problems and solutions in this chapter are different from the following work: 1) Data aggregation [26, 48, 59, 77], which combines the data coming from different sensors that detect common phenomena so as to save transmission energy. Data aggregation techniques often apply simple operations, such as average, max and min, directly on the raw data, and thus are different from the proposed decision aggregation schemes which combine the clustering results from multiple sensor nodes. 2) Multisensor data fusion [32, 93], which gathers and fuses data from multiple sensors in order to achieve higher accuracy. Typically, it combines each sensor's vector of confidence probabilities that the observed target belongs to predefined classes. It is similar to supervised classification, since each confidence probability corresponds to a labeled class. 3) Ensemble classification [19, 27], which combines multiple supervised models or integrates supervised models with unsupervised models for improved classification accuracy. Existing ensemble classification methods cannot be directly applied to our problem setting because they conduct centralized instead of distributed classification and they require sufficient labeled data to train the base models.

3.8 Conclusions

In this chapter, we consider the problem of decision aggregation in sensor networks. We propose two protocols, HAC and cHAC, which work on tree topologies. HAC let each sensor node locally make cluster analysis and forward the decision to its parent node. The decisions are aggregated along the tree, and eventually the global consensus is achieved at the sink node. As an extension of HAC, cHAC can trade energy for accuracy, and thus is able to provide flexible service to various applications.

Chapter 4

Decision Aggregation with Sensor Selection

4.1 Introduction

The aforementioned data and decision aggregation schemes reduce data transmission and resource consumption by "summarizing" the data reported by the sensor nodes. However, in many applications of cyber-physical systems, it is desired that raw sensory data be delivered as is. For example, zoologists may want to use a sensor network to automatically collect high-quality video or audio recordings of wild animals [96, 97]. Moreover, some raw data (e.g., video and audio clips) can help people label the events detected by the sensor nodes. However, the system resources may not be able to afford the delivery of all the raw data. Consequently, we have to select sensors to perform data collection and transmission based on the quality of their captured information.

In cyber-physical systems, the concept of QoI varies in different contexts. We are interested in decision making applications that target on classifying or predicting the current or future state of the physical world through combining the information from the sensor nodes. One representative example is the species classification using the information provided by multiple audio or video sensors [9, 38, 76]. Other examples include target surveillance and recognition, habitat and environmental monitoring, health care or assisted living, etc [2, 14, 30, 44, 45, 60, 62]. In this context, the definition of QoI is motivated by the following two observations:

Observation 1: Consider a set of microphone sensors deployed to record bird vocalizations. Suppose one sensor suffers from circuit board noise, and another is located far away from the birds and close to a group of frogs. Intuitively, the data collected by these two sensors should not be forwarded since they contain substantial noise or are irrelevant to the mission.

Observation 2: To achieve data diversity, we would probably not like both of two camera sensors to upload their data if they always take similar pictures. Instead, we would rather allow only one

of them to transmit, and save the network bandwidth for a microphone sensor which monitors the same objects, despite the audio data is usually not as informative as video data.

The above intuitions lead to the two aspects of QoI: *Reliability* and *Redundancy*. Essentially, reliability implies the degree to which each individual sensor node contributes to the classification task, while redundancy represents the information overlap among different sensor nodes. In this chapter, we set our goal as providing a data selection and transmission service for decision making applications of sensor networks that can optimize QoI, namely, maximize the reliability of sensory data while eliminating their redundancies, under the constraint of network resources. Achieving this, however, is challenging in cyber-physical systems, due to the problems listed below.

- In cyber-physical systems, the sensory data are distributed over a large number of sensor nodes. Furthermore, the system resources cannot afford the delivery of all the raw data, otherwise there is no need to conduct data selection. This eliminates the applicability of centralized solutions working directly on the raw data.

- The reliability and redundancy of a sensor node reflect its relation to the other nodes, and thus cannot be estimated in isolation. Therefore, without obtaining the information from all the sensor nodes, it is hard to precisely estimate the QoI of individual nodes.

- As pointed out in [76], in many applications of cyber-physical systems, the amount of labeled training data is usually small, which can be attributed to the remote, harsh, and sometimes even hostile locales where sensor networks are normally deployed. Without sufficiently large training set, classification algorithms may not be able to describe the characteristics of each class, and thus can potentially make inaccurate predictions on new data.

- The QoI of sensor nodes may be dynamically changing. The dynamics could be resulted from many factors, such as the energy supply of the sensor nodes, the continuous variation of the surveilled environment, and the mobility of the events or even the sensors.

- The data transmission in wireless environment is rather complicated, due to the broadcast nature of wireless communication. How to efficiently utilize wireless spectrum in order to maximize the QoI delivered to the sink remains a problem of great challenge.

The main purpose of this work is to address the above challenges. First of all, we develop a novel online algorithm to estimate the QoI of individual sensor nodes through exploring their clustering results reported to the sink of each mission. Specifically, the reliability of a source node is determined by the level to which its classification or prediction result agrees with those of the majority of other nodes, while the data redundancy between two sensor nodes is measured through investigating the similarity of their clustering results. The algorithm maintains a sliding time window, and the QoI estimates are automatically updated once events within the window are refreshed. Moreover, based on the QoI continuously output by the algorithm, we formulate an optimization framework which aims at maximally utilizing the network resources in order to achieve the optimal aggregate quality of delivered information for a sensor network running multiple concurrent missions. A distributed joint design of data selection and transmission is then proposed to solve this optimization problem.

The rest of the chapter is organized as follows. Section 4.2 provides an overview of the system model and problem formulation. In Section 4.3, we propose the metrics and methods used to estimate QoI. Section 4.4 and Section 4.5 presents our data selection and transmission scheme and its distributed implementation. The proposed scheme is evaluated in Section 4.6. We summarize the related work in Section 4.7. Section 4.8 concludes the chapter.

4.2 System Overview

Consider a sensor network of N nodes which perform M classification missions concurrently. In each mission, there are multiple source nodes that sense the physical surroundings and a single sink node whose task is to store and process the sensory readings. In addition, some relay nodes are deployed to enable data forwarding. The source and sink nodes can also help relay the traffic. A sensor node may be associated with multiple missions simultaneously.

Figure 4.1 shows an illustrative sensor network in which 3 concurrent missions are performed. In this scenario, 5 source nodes, i.e., the shaded nodes $s_1, \ldots, s_5$, collect and forward data to 3 different sink nodes, d_1, d_2, d_3, each of which corresponds to a particular mission. Among the source nodes, s_2, s_3, and s_4 serve for multiple missions. In particular, s_2 serves for mission 1 and 2, s_3 serves for mission 1, 2, and 3, and s_4 serves for mission 2 and 3. Nodes $s_6, \ldots, s_{10}$ work as

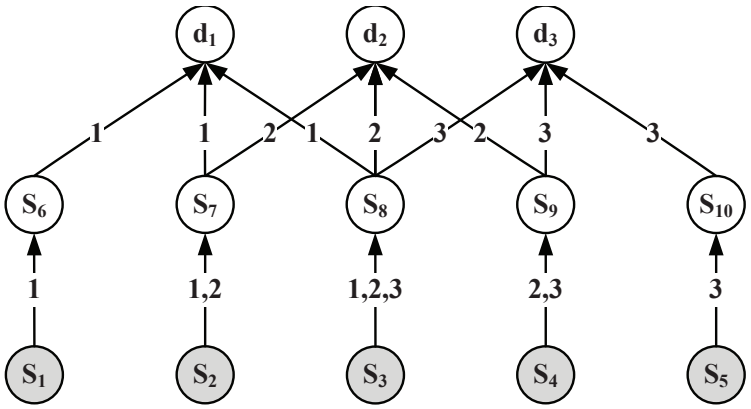


Figure 4.1: An illustrative sensor network in which 3 concurrent missions are performed.

the relay nodes, forwarding data from the source nodes to the sink nodes. The number(s) on each edge indicates the index of the sink node to which the traffic flow on this edge is heading. As can be seen, there may exist flows towards different destinations going through the same edge. For example, the edge connecting node s_3 and s_8 forwards the data to 3 sink nodes since s_3 participates in all the missions.

Based on the above example, we give an overview of the formulation of and solution to the QoI based data selection and transmission problem, which is mathematically formalized as an optimization program $\mathbf{P}$ in Section 4.4.1. The objective of $\mathbf{P}$ is to select a subset of sensor nodes for each mission so that the aggregate reliability of their data can be maximized, under the network resource constraint and the data redundancy constraint. Suppose in this example, the network resource constraint requires that within a time slot, each link can forward data for only one mission. Thus, the source nodes s_2, s_3, and s_4 need to figure out for which mission they should serve. To achieve this, as suggested in Section 4.3.1, each source should send its clustering result of the detected events to the sink node where the reliability and redundancy of its data can be estimated. Particularly, the metric of data reliability is developed in Section 4.3.3 based on the decision aggregation procedure introduced in Section 4.3.2. On the other hand, Section 4.3.4 gives the measure of data redundancy. The QoI estimates are then sent back to the source nodes so that they can locally decide who and for which mission would have a chance to collect data based on a distributed algorithm developed in Section 4.4.2. This algorithm solves $\mathbf{P}$ through decomposing it into a data selection subproblem and a data transmission subproblem that can be

tackled separately. In Section 4.5, we provide a detailed description on how the proposed data selection and transmission scheme is implemented in a distributed manner.

4.3 Quality of Information

This section starts with a brief introduction on how to preprocess sensory data. Then we define the metrics of the data reliability and redundancy, and elaborate on how they are estimated in each mission of the sensor network.

4.3.1 Data Preprocessing

Consider a mission which involves n source nodes, denoted by s_i $(i = 1, 2, ..., n)$. When an event takes place, all the source nodes collect sensory readings about it. Suppose the goal of this mission is to classify the detected events into m different classes. Let $\mathcal{E} = \{e_i |\ i = 1, 2, ..., t\}$ denote the sequence of events (sorted in chronological order) detected by the source nodes. Suppose only a small portion of the events are labeled, and what we need to do is to find out the labels of the rest events. Due to the scarcity of network resources, only a subset of the source nodes can deliver their data to the sink. As previously discussed, without a global view of the information from all the sensor nodes, it is hard to precisely estimate the QoI of each node. A plausible substitution of the raw data is the class labels of the observed events predicted by the sensor nodes. The intuition is as follows. First, if the classification result of a sensor node agrees with those of the majority of others, its data are more likely to be reliable, given the assumption that the majority of the sensor nodes have acceptable classification accuracy. Second, if two sensor nodes always make the same prediction, their information may be redundant.

The challenge of this solution, as aforementioned, is the lack of label information. Without sufficient label information, the classification results of individual sensors cannot be accurate. To tackle this problem, we suggest that each source node locally conduct cluster analysis, which groups data points only based on the similarity of their feature values without any training. The clustering results can provide useful constraints for the task of classification when the labeled data is insufficient, since the data that have similar feature values are usually more likely to share the same class label. Towards this end, we let each of the n source nodes deliver to the sink its clustering

result, in which the events in $\mathcal{E}$ are partitioned into m clusters. Thus, there are totally $l = mn$ different clusters generated by the source nodes, denoted by $c_j,\ j = 1, 2, ..., l$. With the clustering results as well as the label information, the sink is now able to estimate the QoI of each source node. The estimation is based on the *Decision Aggregation* procedure proposed in our previous work [76]. We will first provide a brief introduction of this procedure in the next subsection, and then explain in detail how the QoI of each source node is derived accordingly in the rest of this section.

4.3.2 Decision Aggregation

The decision aggregation procedure takes as input the clustering results of multiple sensors as well as the label information, and outputs a class label for each event. It first models the relationship between the events and the input clusters as a bipartite graph, called *belief graph*. In belief graph, each input cluster links to the events it contains. Moreover, to integrate label information into the belief graph, one more set of vertices are added to represent the labels of the events. The labeled events are then connected to the corresponding label vertices. Figure 4.2 provides an example of belief graph involving $n = 3$ sensor nodes and $t = 10$ events. In this case, suppose the mission is to classify the events into $m = 2$ different classes, then there are totally $l = mn = 6$ different clusters.

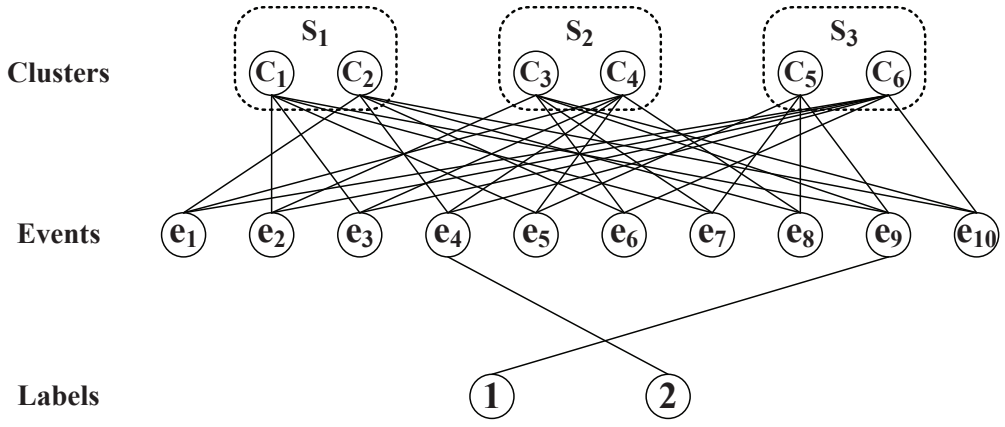


Figure 4.2: An example of belief graph

The belief graph can be represented by two adjacency matrices: (i) Clustering matrix $A = (a_{ij})_{t \times l}$, where a_{ij} indicates whether event e_i is assigned to cluster c_j. (ii) Groundtruth matrix

$Z = (z_{ik})_{t \times m}$, where z_{ik} denotes whether e_i's observed label is k. Then, two sets of probability vectors are defined. First, each event e_i is associated with a m-dimensional probability vector, denoted by $\vec{x}_{i\cdot} = (x_{ik})$. Each element of $\vec{x}_{i\cdot}$, say x_{ik}, indicates the probability of e_i belonging to the k-th class. Second, for each input cluster c_j, a m-dimensional probability vector, denoted by $\vec{y}_{j\cdot} = (y_{jk})$, is also defined. Each element of this vector is the probability that the majority of the events contained in c_j are assigned to a particular class. $\vec{x}_{i\cdot}$ and $\vec{y}_{j\cdot}$ work as the variables in the optimization program solving the decision aggregation problem:

$$\begin{aligned} \textbf{DA}: \min \quad & \sum_{i=1}^{t}\sum_{j=1}^{l} a_{ij}||\vec{x}_{i\cdot} - \vec{y}_{j\cdot}||^2 + \alpha \sum_{i=1}^{t} b_i||\vec{x}_{i\cdot} - \vec{z}_{i\cdot}||^2 \\ \text{s.t.} \quad & \vec{x}_{i\cdot} \geq \vec{0}, \quad |\vec{x}_{i\cdot}| = 1 \quad \text{for} \quad i = 1, 2, ..., t \\ & \vec{y}_{j\cdot} \geq \vec{0}, \quad |\vec{y}_{j\cdot}| = 1 \quad \text{for} \quad j = 1, 2, ..., l \end{aligned}$$

where $||.||$ and $|.|$ denote a vector's L2 and L1 norm respectively. Besides, $b_i = \sum_{k=1}^{m} z_{ik}$ is a flag variable indicating whether e_i is labeled or not, and α is a predefined parameter. **DA** is actually a convex program, which makes it possible to find a global optimal solution. To achieve consensus among the clustering results of multiple sensor nodes, **DA** aims at finding the optimal probability vectors of the event nodes ($\vec{x}_{i\cdot}$) and the cluster nodes ($\vec{y}_{j\cdot}$) that can minimize the disagreement over the belief graph, and in the meanwhile, comply with the label information. Moreover, since $\vec{x}_{i\cdot}$ and $\vec{y}_{j\cdot}$ are probability vectors, each of their components must be greater than or equal to 0 and the sum should equal 1.

4.3.3 Data Reliability

As previously discussed, the reliability of a source node can be measured by the level to which its classification or prediction result agrees with those of the majority of other nodes. This can be inferred from the solution of **DA**. In its objective function, the first term ensures that an input cluster has similar probability vector as the events it contains, namely, $\vec{x}_{i\cdot}$ should be close to $\vec{y}_{j\cdot}$ if event e_i is connected to cluster c_j in the belief graph. Let's put it in a more straightforward way. If we fix the values of $\vec{x}_{i\cdot}$ as constants, then the objective function becomes a convex function with respect to $\vec{y}_{j\cdot}$. Its minimum can be obtained by setting the partial derivatives $\frac{\partial f(X,Y)}{\partial y_{jk}}$, ($k =$

$1, 2, \ldots, m$) to 0:

$$\vec{y}_{j\cdot} = \frac{\sum_{i=1}^{t} a_{ij}\vec{x}_{i\cdot}}{\sum_{i=1}^{t} a_{ij}}. \tag{4.1}$$

As one can see, $\vec{y}_{j\cdot}$ is actually the average of the probability vectors of the events that belong to cluster c_j. On the other hand, the second term of **DA**'s objective function puts the constraint that a labeled event's probability vector $\vec{x}_{i\cdot}$ should not deviate much from the corresponding groundtruth vector $\vec{z}_{i\cdot}$, and α can be considered as the shadow price payment for violating this constraint. If the values of $\vec{y}_{j\cdot}$ are fixed, the optimal $\vec{x}_{i\cdot}$ can be derived through the following formula:

$$\vec{x}_{i\cdot} = \frac{\sum_{j=1}^{l} a_{ij}\vec{y}_{j\cdot} + \alpha b_i \vec{z}_{i\cdot}}{\sum_{j=1}^{l} a_{ij} + \alpha b_i}. \tag{4.2}$$

For an unlabeled event e_i, since its flag variable $b_i = 0$, $\vec{x}_{i\cdot}$ is calculated through averaging the probability vectors of the clusters containing e_i. If e_i is labeled, $\vec{x}_{i\cdot}$ becomes the weighted average of clustering information and label information tuned by the shadow price α. According to Eqn. (4.2), given that the majority of the sensor nodes are collecting data with acceptable quality, the probability vectors of most of the events should be close to the groundtruth, since the errors of individual sensors can be canceled out by the averaging operation. In other words, x_{ik} should be the largest element of $\vec{x}_{i\cdot}$ if the groundtruth label of e_i is k. Consequently, by Eqn. (4.1), the elements of $\vec{y}_{j\cdot}$ will be skewed if the majority of the events contained in cluster c_j belong to the same class in groundtruth. In contrast, if c_j's events have diversified groundtruth labels, $\vec{y}_{j\cdot}$'s elements should be evenly distributed.

Table 4.1: Probability Vectors of Events

Event	e_1	e_2	e_3	e_4	e_5
Vector	$\vec{x}_{1\cdot}$	$\vec{x}_{2\cdot}$	$\vec{x}_{3\cdot}$	$\vec{x}_{4\cdot}$	$\vec{x}_{5\cdot}$
Value	(1,0)	(1,0)	(1,0)	(0,1)	(1,0)
Event	e_6	e_7	e_8	e_9	e_{10}
Vector	$\vec{x}_{6\cdot}$	$\vec{x}_{7\cdot}$	$\vec{x}_{8\cdot}$	$\vec{x}_{9\cdot}$	$\vec{x}_{10\cdot}$
Value	(0,1)	(0,1)	(0,1)	(1,0)	(0,1)

An illustrative example may shed more light on this point. Suppose the optimal probability vectors of the events in Fig. 4.2 are listed in Table 4.1. For the sake of simplicity, in this case we

set the elements of $\vec{x}_{i\cdot}$ as binaries. In reality, they cannot be simply 0 or 1, but rather decimal numbers between 0 and 1. For cluster c_1, its probability vector can be derived following Eqn. (4.1):

$$\vec{y}_{1\cdot} = (\vec{x}_{2\cdot} + \vec{x}_{3\cdot} + \vec{x}_{5\cdot} + \vec{x}_{7\cdot} + \vec{x}_{9\cdot})/5 = (0.8, 0.2).$$

$\vec{y}_{1\cdot}$ is quite skewed since 4 of the 5 events in c_1 have a probability vector of (1,0). Similarly, the probability vectors of other clusters are calculated and shown in Table 4.2. As one can see, some clusters (e.g., c_5 and c_6) have uniform probabilities since they have equal number of events with vector (1,0) and (0,1).

Table 4.2: Cluster and Sensor Entropies

Cluster	c_1	c_2	c_3	c_4	c_5	c_6
Vector	$\vec{y}_{1\cdot}$	$\vec{y}_{2\cdot}$	$\vec{y}_{3\cdot}$	$\vec{y}_{4\cdot}$	$\vec{y}_{5\cdot}$	$\vec{y}_{6\cdot}$
Value	(0.8,0.2)	(0.2,0.8)	(0.6,0.4)	(0.4,0.6)	(0.5,0.5)	(0.5,0.5)
Entropy	0.7219	0.7219	0.9710	0.9710	1	1

Sensor	s_1	s_2	s_3
Entropy	0.7219	0.9710	1
Reliability	0.2781	0.0290	0

Guided by the above observation, we propose to estimate the reliability of a source node based on the degree of impurity of the clusters it generates. The smaller the degree of impurity, the more skewed the probability vector. In the above example, c_1 has a pretty low degree of impurity, while c_5 and c_6 are the most impure clusters. In this chapter, we use entropy [17], a quantitative representation of random variable uncertainties in information theory, to measure the impurity of clusters. For a cluster c_j, its entropy can be calculated as $-\sum_{k=1}^{m} y_{jk} \log_2 y_{jk}$. The entropy of a source node s_i is defined as the weighted average of its clusters' entropies:

$$-\sum_{c_j \in s_i} \omega_j \sum_{k=1}^{m} y_{jk} \log_2 y_{jk},$$

where the weight ω_j amounts to the ratio of the number of events in c_j over the total event number. The entropies of the six clusters as well as the three sensor nodes are shown in Table 4.2. As one can see, the entropy values can precisely reflect the degree of impurity. However, entropy

cannot be directly used as the estimate of reliability. This is simply because larger entropy means higher impurity, which reversely implies less reliability. According to the principle of maximum entropy [17], the largest entropy of a sensor node is reached when the probability vectors of its clusters have equal elements, which are $\frac{1}{m}$. Thus, the largest entropy is $-\log_2(\frac{1}{m}) = \log_2(m)$. Subtracting the entropy of each node from this number gives the reliability estimate of this node:

$$\log_2(m) + \sum_{c_j \in s_i} \omega_j \sum_{k=1}^{m} y_{jk} \log_2 y_{jk}. \tag{4.3}$$

The reliability estimates of the three nodes are shown in Table 4.2. As the numbers suggest, now the sensors with impure clusters have lower reliability scores.

In contrast to decision aggregation which works offline on all the collected data, sensor selection calls for a online mechanism which can adaptively determine the set of sensor nodes to transmit based on their recent data's quality of information. The first step to this end is a novel algorithm called Incremental Reliability Estimation (IRE). The IRE algorithm applies a sliding time window of width w to the set of sequentially occurred events $\mathcal{E} = \{e_i |\ i = 1, 2, ..., t, ...\}$, and incrementally outputs the reliability estimate of each sensor node according to the sensory data of the events within the window. Upon the occurrence of v $(1 \leq v \leq w)$ new events, the window slides, with the v new events (denoted by $e_{t+i},\ i = 1, ..., v$) added and v outdated events (denoted by $e_{t-w+i},\ i = 1, ..., v$) removed. On the other hand, each of the source nodes updates its clustering result through either re-clustering or incremental clustering [4]. The clustering results of the newly arrived data are then reported to the sink node of each mission where the IRE algorithm is invoked.

The basic idea of the IRE algorithm is as follows. It first calculates the probability vectors of the newly-occurred events through Eqn. (4.2), and then uses them to update the probability vectors of clusters according to an incremental version of Eqn. (4.1) as below:

$$\vec{y}_{j\cdot} = \frac{\sum_{i=t-w+1}^{t} a_{ij}\vec{x}_{i\cdot} - \sum_{i=t-w+1}^{t-w+v} a_{ij}\vec{x}_{i\cdot} + \sum_{i=t+1}^{t+v} a_{ij}\vec{x}_{i\cdot}}{\sum_{i=t-w+1}^{t} a_{ij} - \sum_{i=t-w+1}^{t-w+v} a_{ij} + \sum_{i=t+1}^{t+v} a_{ij}}. \tag{4.4}$$

As can be seen, both the numerator and denominator contain the information from three parts: (i) the events in the original window $(e_{t-w+1}, \ldots, e_t)$ (ii) outdated events $(e_{t-w+1}, \ldots, e_{t-w+v})$ (iii) new events $(e_{t+1}, \ldots, e_{t+v})$. Therefore, the update can be done incrementally through substituting

outdated information with new information, without the need of a complete recalculation. Finally, the updated $\vec{y}_{j\cdot}$ are used as the input of Eqn. (4.3) to derive the reliability score of each sensor node.

In practice, the above computations are conducted via matrix operations, and thus we introduce some matrix notations. Putting the probability vectors of the events in the original window together, we get a probability matrix $X^{(t)}_{w\times m} = (\vec{x}_{(t-w+1)\cdot}, \ldots, \vec{x}_{t\cdot})^T$. Similarly, as shown in Eqn. (4.5), we define the probability matrices of the outdated events ($X^{\text{old}}_{v\times m}$), the newly-occurred events ($X^{\text{new}}_{v\times m}$), as well as the events in the updated window ($X^{(t+v)}_{w\times m}$).

$$\overbrace{\underbrace{\vec{x}_{(t-w+1)\cdot}, \ldots, \vec{x}_{(t-w+v)\cdot}}_{X^{\text{old}}} \mid \underbrace{\vec{x}_{(t-w+v+1)\cdot}, \ldots, \vec{x}_{t\cdot}}_{X^{(t+v)}}}^{X^{(t)}} \mid \overbrace{\vec{x}_{t+1\cdot}, \ldots, \vec{x}_{t+v\cdot}}^{X^{\text{new}}} \tag{4.5}$$

Then, we generate a clustering matrix $A = (a_{ij})$ (recall that a_{ij} indicates whether event e_i is assigned to cluster c_j.) corresponding to each of the above probability matrices. They are denoted by $A^{(t)}$ ($i = t-w+1, \ldots, t$), A^{old} ($i = t-w+1, \ldots, t-w+v$), A^{new} ($i = t+1, \ldots, t+v$), $A^{(t+v)}$ ($i = t-w+v+1, \ldots, t+v$), respectively. In the following notation definitions, we will use the same correspondence between the superscripts (i.e., "new", "old", "(t)", "$(t+v)$") and the range of event index i.

The detailed steps of IRE are shown in Algorithm 4. Line 8 displays the derivation of X^{new} through Eqn. (4.2), where $B = \text{diag}\big\{(b_i)\big\}$ and $C = \text{diag}\big\{(\sum_{j=1}^{l} a_{ij})\big\}$. Note that the superscript "new" implies $i = t+1, \ldots, t+v$. Additionally, here $Y^{(t)}_{l\times m} = (\vec{y}^{(t)}_{1\cdot}, \ldots, \vec{y}^{(t)}_{l\cdot})^T$ is the probability matrix of all the clusters at time t. The matrix form of Eqn. (4.4) is given at line 9. In this equation, $D^{(t+v)}$ and $F^{(t)}$ are constant matrices. In particular, $D = \text{diag}\big\{(\sum_i a_{ij})\big\}$ corresponds to the items in the denominator of Eqn. (4.4), and $D^{(t+v)}$ is calculated at line 2. $F^{(t)}$ represents the information of remaining events in the time window, and is generated at line 3. In the while loop from line 6 to line 9. X^{new} and $Y^{(t+v)}$ are repeatedly updated by each other, until no notable change occurs at $Y^{(t+v)}$. The convergence is guaranteed by the theory of coordinate descent [5], due to the convexity of **DA**.

At line 10, the derived X^{new} is put into a function of label$(\cdot)$, which converts each probability vector of X^{new} into a binary vector. Precisely, it sets the largest element of a probability vector

Algorithm 4 Incremental Reliability Estimation

Input: The clustering results of the sensor nodes for the v new events, associated with their labels;
Output: The estimate of each node's reliability;

1: Generate A^{new}, B^{new}, C^{new}, D^{new}, and Z^{new}.
2: $D^{(t+v)} \leftarrow D^{(t)} - D^{\text{old}} + D^{\text{new}}$
3: $F^{(t)} \leftarrow {A^{(t)}}^T X^{(t)} - {A^{\text{old}}}^T X^{\text{old}}$
4: $Y^{\text{temp}} \leftarrow Y^{(t)}$
5: Initialize $Y^{(t+v)}$ randomly.
6: **while** $||Y^{(t+v)} - Y^{\text{temp}}|| > \epsilon$ **do**
7: $\quad Y^{\text{temp}} \leftarrow Y^{(t+v)}$
8: $\quad X^{\text{new}} \leftarrow (C^{\text{new}} + \alpha B^{\text{new}})^{-1}(A^{\text{new}} Y^{(t)} + \alpha B^{\text{new}} Z^{\text{new}})$
9: $\quad Y^{(t+v)} \leftarrow {D^{(t+v)}}^{-1}(F^{(t)} + {A^{\text{new}}}^T X^{\text{new}})$
10: **end while**
11: $\widetilde{X}^{\text{new}} \leftarrow \text{label}(X^{\text{new}})$
12: $\widetilde{Y}^{(t+v)} \leftarrow {D^{(t+v)}}^{-1}({A^{(t)}}^T \widetilde{X}^{(t)} - {A^{\text{old}}}^T \widetilde{X}^{\text{old}} + {A^{\text{new}}}^T \widetilde{X}^{\text{new}})$
13: $\vec{e}_{\text{cluster}}^{(t+v)} = -\text{sum}(\widetilde{Y}^{(t+v)} .* \log_2(\widetilde{Y}^{(t+v)}), 2)$
14: $\vec{\omega}_{\text{cluster}}^{(t+v)} = \text{sum}({A^{(t+v)}}^T, 2)/w$
15: $\vec{p}^{(t+v)} = \log_2(m) - \text{sum}(\text{vec2mat}(\vec{\omega}_{\text{cluster}}^{(t+v)} .* \vec{e}_{\text{cluster}}^{(t+v)}, m), 2)$
16: **return** $\vec{p}^{(t+v)}$

to be 1, and other elements to be 0. The modified matrix $\widetilde{X}^{\text{new}}$ is further used to calculate the final probability vector of clusters $\widetilde{Y}^{(t+v)}$ at line 11. With $\widetilde{Y}^{(t+v)}$, the algorithm is able to derive the estimate of reliability. It first calculates the entropy as well as weight of each cluster at line 12 and 13, respectively. Here $.*$ denotes the operation of element-wise multiplication between two matrices, while the function sum(A,d) sums along the dimension of A specified by scalar d. Finally, the reliability estimates of the sensor nodes are determined according to Eqn. (4.3) via a matlab function vec2mat($\vec{v}$,m) that converts the vector $\vec{v}$ into a matrix with m columns.

4.3.4 Data Redundancy

The data redundancy between two sensor nodes can be measured through investigating the similarity of their clustering results. The comparison of clustering results can be achieved using *Similarity Matrix* [80]. The similarity matrix of a sensor node is defined as $M = (m_{ij})$, where m_{ij} equals 1 if event e_i and e_j are put into the same cluster by this node, and 0 otherwise.

The similarity matrices of sensor s_1 and s_2 in the previous example for the first 5 events are shown in Table 4.3 and Table 4.4, respectively. To quantitatively measure the difference between two similarity matrices M_1 and M_2, four numbers are defined as follows:

Table 4.3: Similarity Matrix of s_1

Event	e_1	e_2	e_3	e_4	e_5
e_1	1	0	0	1	0
e_2	0	1	1	0	1
e_3	0	1	1	0	1
e_4	1	0	0	1	0
e_5	0	1	1	0	1

Table 4.4: Similarity Matrix of s_2

Event	e_1	e_2	e_3	e_4	e_5
e_1	1	0	1	1	1
e_2	0	1	0	0	0
e_3	1	0	1	1	1
e_4	1	0	1	1	1
e_5	1	0	1	1	1

- f_{00}: number of event pairs belonging to different clusters in both M_1 and M_2.
- f_{01}: number of event pairs belonging to different clusters in M_1 and the same cluster in M_2.
- f_{10}: number of event pairs belonging to the same cluster in M_1 and different clusters in M_2.
- f_{11}: number of event pairs belonging to the same cluster in both M_1 and M_2.

Two measures based on the above quantities are widely used: Rand statistic and Jaccard coefficient [80]. Their definitions are shown below:

$$\text{Rand} = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}} \qquad \text{Jaccard} = \frac{f_{11}}{f_{01} + f_{10} + f_{11}} \tag{4.6}$$

For the above two matrices, the quantities are $f_{00} = 2$, $f_{01} = 4$, $f_{10} = 2$, and $f_{11} = 2$. Thus, the Rand statistic is $(2 + 2)/10 = 0.4$, while the Jaccard coefficient is $2/(4 + 2 + 2) = 0.25$. In reality, there is no need to exhaustively check each pair of the sensor nodes to find redundancy. By adding some simple rules, the searching space can be dramatically reduced. For example, we may only need to compare the clustering results of the nodes in close proximity and equipped with the same types of sensors. Moreover, the sliding window limits the size of the similarity matrices. To further mitigate the storage and computation overhead, we can randomly sample the elements of a matrix instead of maintaining its entirety. Additionally, at each time when the window slides, for each matrix only the entries involving the newly-occurred events are updated, with others remaining unchanged.

4.4 Data Selection and Transmission

With the previously defined Quality of Information, we are now ready to formally formulate the QoI based data selection and transmission problem, which is denoted by **P**. In this section, we will first introduce the objective function as well as the constraints of **P**, and then propose a distributed algorithm to solve it. The notations defined in this section may have been used in preceding parts of the chapter. However, they can be easily distinguished from the context, and thus we believe no confusion would occur.

4.4.1 Problem Formulation

Objective Function

The objective function of **P** is the aggregate data reliability of all the missions. In this formulation, we assume that the information collected in different missions are independent of one another. To simplify the presentation, we assume the reliability measures for different missions are normalized and of the same scale. One can easily prioritize the missions by associating the reliability of each mission with a weight parameter. Within each mission, after removing the information overlap among different sensor nodes via the data redundancy constraint, the data reliability of a sensor set can be approximated as the summation of individual sensors' reliability. Guided by this intuition, we specify the objective function as $\sum_{k=1}^{M}\sum_{i=1}^{N} p_i^k x_i^k$. In this function, p_i^k is the reliability estimate of the sensor node s_i with regard to the k-th mission. $p_i^k = 0$ if s_i is not a source node of mission k. $x_i^k \in \{0, 1\}$ is a variable indicating whether sensor s_i is selected to collect data for mission k.

Network Resource Constraint

The network resources could be bandwidth, energy, storage, and many others. For the sake of simplicity, in this chapter we focus on bandwidth, which may be the most difficult one to handle. The proposed framework, however, can be easily extended to other resources. In wireless networks, the neighboring links may contend for bandwidth due to the broadcast nature of wireless transmission. The contention relations among the links can be captured by a conflict graph [41], based on the network topology. In the conflict graph, each vertex represents a link, and an edge between two vertices implies the contention between the two corresponding links, i.e., they cannot transmit at the same time. Given a conflict graph, we can identify all its independent sets of vertices that have

no edges between each other. The links in an independent set can transmit simultaneously.

Suppose $\mathcal{L}$ is the set of all the links in the sensor network, and let $\mathcal{I}$ denote the collection of independent sets. We represent an independent set, I_q $(q = 1, 2, ..., |\mathcal{I}|)$, as a $|\mathcal{L}|$-dimensional bandwidth vector, which is c^q. In c^q, an element $c^q_{i,j} = b_{i,j}$ if $(i,j) \in I_q$ and 0 otherwise, where $b_{i,j}$ denotes the bandwidth capacity of link $(i,j) \in \mathcal{L}$. The feasible bandwidth region Π at the link layer is defined as the convex hull of these vectors:

$$\Pi := \{c \mid c = \sum_{q=1}^{|\mathcal{I}|} \alpha_q c^q,\ \alpha_q \geq 0,\ \sum_{q=1}^{|\mathcal{I}|} \alpha_q = 1\}.$$

Let $c^k_{i,j}$ denote the amount of bandwidth of link (i,j) allocated to the flow of mission k. Then $c_{i,j} = \sum_{k=1}^{M} c^k_{i,j}$ is the aggregate bandwidth of link (i,j). According to the above definition, the bandwidth vector of all the links $c = (c_{i,j})$ should satisfy $c \in \Pi$. Suppose r^k_i is the data collection rate of source node s_i for mission k. r^k_i implies s_i's demand for bandwidth, and $r^k_i = 0$ if s_i is not a source of mission k. Once s_i is allowed to collect data, to prevent its queue from overflow, for each mission the summation of the bandwidth allocated to s_i's incoming flows and its demanded bandwidth should not exceed the aggregate bandwidth for its outgoing flows. This motivates the network resource constraint for bandwidth resource:

$$r^k_i x^k_i \leq \sum_{j:(i,j)\in\mathcal{L}} c^k_{i,j} - \sum_{j:(j,i)\in\mathcal{L}} c^k_{j,i}.$$

Data Redundancy Constraint

In Section 4.3.4, we discuss the redundancy measures of sensor pairs. By setting up a threshold for the redundancy measure, we can identify the redundant node pairs and thus effectively control the redundancy level in the network. To capture the redundancy relations among the sensor nodes, we first take a graph expansion on the network topology. In particular, for each source node s_i, if it serves for mission k, we create a virtual node denoted by $s_{i,k}$. $s_{i,k}$ is connected to s_i through a virtual link with a capacity of r^k_i and works as a virtual source of mission k. In $\mathbf{P}$, each virtual source $s_{i,k}$ is associated with a previously defined indicator variable $x_{i,k}$.

Figure 4.3 shows the expanded topology of the sensor network in Fig. 4.1. In this case, we simply assume that in each mission all the source nodes are redundant with each other. For

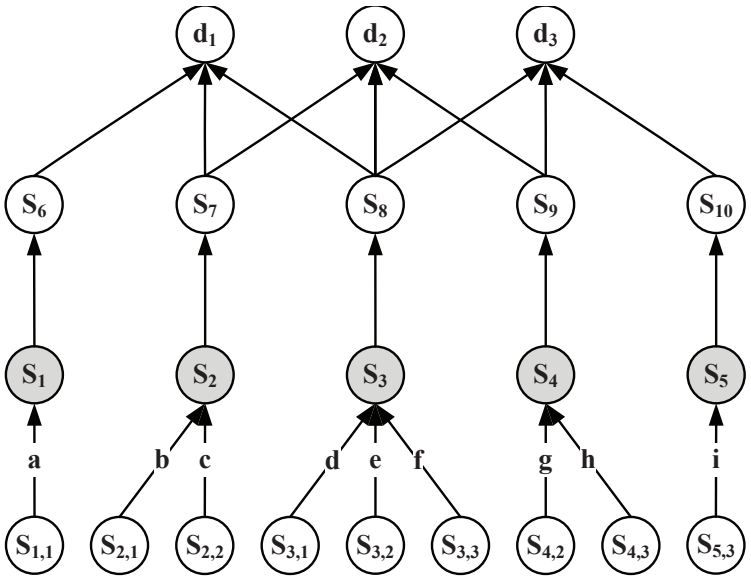


Figure 4.3: Expanded topology

example, the source node s_1, s_2, and s_3 are redundant in mission 1, and thus they should not collect data simultaneously for mission 1. Equivalently, in the expanded topology this implies that the virtual link a, b, and d cannot transmit at the same time. This kind of confliction among the virtual links can be modeled by constructing a bipartite graph called data redundancy graph. As shown in Fig. 4.4, in the data redundancy graph a set of auxiliary nodes (black nodes) are created and connected to the virtual source nodes that are redundant with one another. With this graph transformation, the problem of finding a subset of source nodes which can simultaneously collect data becomes the problem of identifying a matching, i.e., a set of links without common nodes, of the data redundancy graph.

The virtual nodes connecting to the same source may also have conflicts, since the sensing devices on a node may not be able to serve multiple missions at the same time. To tackle this problem, we build another bipartite graph called sensor conflict graph. Similar to the data redundancy graph, the conflicting virtual nodes are connected to the same auxiliary nodes as drawn in Fig. 4.4. Again, matching algorithms can be used to find the conflict-free node set. Similar to what we did when formulating the network resource constraint, the feasible bandwidth region of the virtual links can be derived through combining the bandwidth vectors of all the matchings in the data redundancy graph and sensor conflict graph.

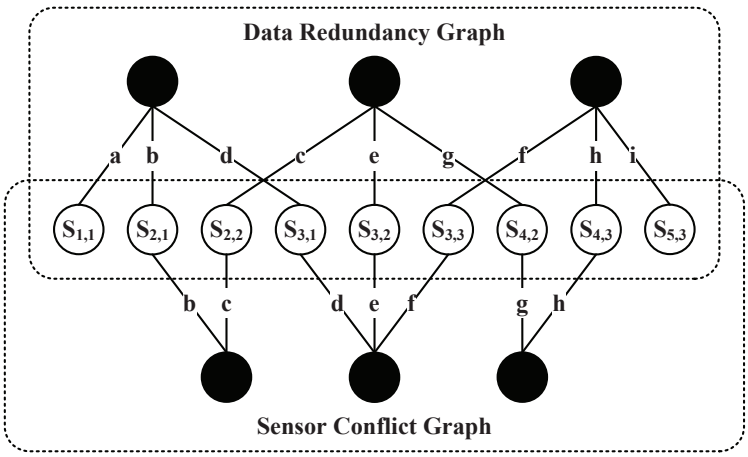


Figure 4.4: Data redundancy graph and sensor conflict graph

Putting the above objective function and constraints together gives us the complete optimization program:

$$\mathbf{P}: \max \quad \sum_{k=1}^{M}\sum_{i=1}^{N} p_i^k x_i^k \tag{4.7}$$

$$\text{subject to} \quad r_i^k x_i^k \leq \sum_{j:(i,j)\in\mathcal{L}} c_{i,j}^k - \sum_{j:(j,i)\in\mathcal{L}} c_{j,i}^k \tag{4.8}$$

$$c \in \Pi$$

The data redundancy constraint is not explicitly present in $\mathbf{P}$, since it has already been captured in Π which covers the feasible bandwidth region of not only the real links but also the virtual links.

4.4.2 Distributed Algorithm via Dual Decomposition

$\mathbf{P}$ is actually a mixed integer program since the feasible values of $x = (x_i^k)$ are restricted to be 0 or 1, making it difficult to find the optimal solution. Furthermore, solving $\mathbf{P}$ directly requires global coordination of all the nodes, which is impractical in a distributed environment such as sensor networks. To address these challenges, we first relax $\mathbf{P}$ into a convex problem, and propose a distributed solution through dual decomposition.

Convex Relaxation and Dual Decomposition

By allowing x to take any value between 0 and 1, $\mathbf{P}$ can be relaxed into a convex program, denoted as $\widetilde{\mathbf{P}}$. Due to the convexity of $\widetilde{\mathbf{P}}$, strong duality can be achieved (Chapter 5.2.3 in [6]). Therefore,

there exists a unique maximizer (x^*, c^*) for $\widetilde{\mathbf{P}}$, which can be attained by a distributed algorithm derived via formulating and solving the Lagrange dual problem of $\widetilde{\mathbf{P}}$. In order to achieve this, we first take a look at the Lagrangian of $\widetilde{\mathbf{P}}$:

$$L(x, c, \mu) = \sum_{k=1}^{M}\sum_{i=1}^{N} p_i^k x_i^k - \sum_{k=1}^{M}\sum_{i=1}^{N} \mu_i^k (r_i^k x_i^k - \sum_{j:(i,j)\in\mathcal{L}} c_{i,j}^k + \sum_{j:(j,i)\in\mathcal{L}} c_{j,i}^k).$$

In $L(x, c, \mu)$, $\mu = (\mu_i^k)$ is the vector of Lagrangian multipliers, corresponding to the network resource constraint (Eqn. (4.8)). μ_i^k is also interpreted as the "shadow price" of the constraint, which can be understood as the "cost" a node will be charged if it violates the constraint. Furthermore, since

$$\sum_{k=1}^{M}\sum_{i=1}^{N} \mu_i^k (\sum_{j:(i,j)\in\mathcal{L}} c_{i,j}^k - \sum_{j:(j,i)\in\mathcal{L}} c_{j,i}^k) = \sum_{k=1}^{M}\sum_{i=1}^{N}\sum_{j:(i,j)\in\mathcal{L}} c_{i,j}^k(\mu_i^k - \mu_j^k),$$

we reorganize the Lagrangian as follows:

$$\begin{aligned} L(x, c, \mu) &= \sum_{k=1}^{M}\sum_{i=1}^{N} p_i^k x_i^k - \sum_{k=1}^{M}\sum_{i=1}^{N} \mu_i^k r_i^k x_i^k + \sum_{k=1}^{M}\sum_{i=1}^{N} \mu_i^k (\sum_{j:(i,j)\in\mathcal{L}} c_{i,j}^k - \sum_{j:(j,i)\in\mathcal{L}} c_{j,i}^k) \\ &= \sum_{k=1}^{M}\sum_{i=1}^{N} (p_i^k - \mu_i^k r_i^k) x_i^k + \sum_{k=1}^{M}\sum_{i=1}^{N}\sum_{j:(i,j)\in\mathcal{L}} c_{i,j}^k(\mu_i^k - \mu_j^k). \end{aligned}$$

The dual of the primal problem $\widetilde{\mathbf{P}}$ is:

$$\mathbf{D}: \min_{\mu\geq 0} \quad D(\mu),$$

where the dual objective function $D(\mu)$ is given as

$$D(\mu) := \max_{x\in X,\ c\in\Pi} \quad L(x, c, \mu).$$

In the dual objective function, the Lagrangian multiplier (shadow price) μ serves as the dual variable. Furthermore, $D(\mu)$ can be decomposed into two separate optimization problems: $D(\mu) =$

$D_1(\mu) + D_2(\mu)$. $D_1(\mu)$ and $D_2(\mu)$ are defined below:

$$D_1(\mu) := \max_{x \in X} \quad \sum_{k=1}^{M} \sum_{i=1}^{N} (p_i^k - \mu_i^k r_i^k) x_i^k$$

$$D_2(\mu) := \max_{c \in \Pi} \quad \sum_{k=1}^{M} \sum_{i=1}^{N} \sum_{j:(i,j) \in \mathcal{L}} c_{i,j}^k (\mu_i^k - \mu_j^k)$$

Among them, $D_1(\mu)$ denotes the *data selection problem*, while $D_2(\mu)$ is the *data transmission problem*. In particular, the data selection problem aims at finding the subset of source nodes whose data have the maximum aggregate data reliability, while the data transmission problem aims at scheduling the transmission of the sensory data picked by the data selection problem. In the rest of this section, we will first elaborate on these two problems separately, and then explain how to develop a distributed joint design of them.

The Data Selection Problem

The data selection problem can be further transformed as follows:

$$\begin{aligned} D_1(\mu) &= \sum_{k=1}^{M} \sum_{i=1}^{N} \max_{0 \le x_i^k \le 1} \Phi(x_i^k) \\ &= \sum_{k=1}^{M} \sum_{i=1}^{N} \max_{0 \le x_i^k \le 1} (p_i^k - \mu_i^k r_i^k) x_i^k. \end{aligned}$$

In other words, the data selection problem can be solved through separately solving the optimization problem of each source node. since $\frac{\mathrm{d}\Phi(x_i^k)}{\mathrm{d}x_i^k} = p_i^k - \mu_i^k r_i^k$ is a constant, once the value of μ is assigned, the optimal value of x_i^k can be calculated as below:

$$x_i^{k*}(\mu) = \arg \max_{0 \le x_i^k \le 1} \Phi(x_i^k) = \begin{cases} 1 & \text{if } p_i^k > \mu_i^k r_i^k \\ 0 & \text{if } p_i^k \le \mu_i^k r_i^k \end{cases}. \tag{4.9}$$

The result is rather interesting, since x_i^k attains optimum at either 0 or 1, even though we have relaxed its feasible range to be any value between 0 and 1. Therefore, we can directly use x_i^{k*} as the solution to **P** without taking the rounding step.

The Data Transmission Problem

We transform the data transmission problem as:

$$\begin{aligned} D_2(\mu) &= \max_{c\in\Pi} \sum_{k=1}^{M}\sum_{i=1}^{N}\sum_{j:(i,j)\in\mathcal{L}} c_{i,j}^k(\mu_i^k-\mu_j^k) \\ &= \max_{c\in\Pi} \sum_{(i,j)\in\mathcal{L}} c_{i,j} \max_{1\le k\le M}(\mu_i^k-\mu_j^k), \end{aligned}$$

which can be solved through a joint design of routing and scheduling.

Routing: For each link (i,j), we find the mission k^* that maximizes $\mu_i^k-\mu_j^k$. Then, at the next time slot, the link (i,j) will be dedicated to forward mission k^*'s data.

Scheduling: Let $w_{i,j}=\mu_i^{k^*}-\mu_j^{k^*}$, we target on choosing a bandwidth vector $c^*=(c_{ij}^*)$ such that:

$$c^* = \max_{c\in\Pi} \sum_{(i,j)\in\mathcal{L}} w_{i,j}c_{i,j}. \tag{4.10}$$

This is actually a linear programming problem, and thus the maximizer can be always found at an extreme point. An extreme point maximizer corresponds to a maximal independent set of the conflict graph. Therefore, this problem is equivalent to the maximum weighted independent set problem over the conflict graph, which is NP-hard. Actually, the conflict graph depends on the underlying interference model. In this chapter, we consider node-exclusive interference model, i.e., links that share a common node cannot transmit or receive simultaneously. This model has been widely used in existing work [12, 53, 77] on network utility maximization. With the node exclusive interference model, the scheduling problem can be reduced to the maximum weighted matching problem, which is polynomial-time solvable. However, the existing polynomial-time solution [64] requires centralized implementation. In [35], a simple distributed approximate algorithm is presented, which is at most a factor of 2 away from the maximum, and has a linear running time $O(|\mathcal{L}|)$. We utilize this algorithm to solve the scheduling problem in a distributed manner.

Actually, the strategy proposed in this chapter is a general framework and thus can be extended to other interference models. For any interference model, as long as an appropriate algorithm can be designed to solve the above scheduling problem, it can be integrated with our framework. In addition, the construction of the aforementioned data redundancy graph and sensor conflict graph is independent of interference models. We also use the distributed matching algorithm discussed

above to find the subset of source nodes that can collect data simultaneously.

Subgradient Algorithm

We use subgradient method [72] to minimize the dual objective function $D(\mu)$. Specifically, μ is adjusted in the opposite direction to the subgradient:

$$\begin{aligned}\mu_i^k(t+1) &= \left[\mu_i^k(t) - h(t)\frac{\partial D(\mu)}{\partial \mu_i^k}\right]^+ \\ &= \left[\mu_i^k(t) + h(t)(r_i^k x_i^k(t) - \sum_{j:(i,j)\in\mathcal{L}} c_{i,j}^k(t) + \sum_{j:(j,i)\in\mathcal{L}} c_{j,i}^k(t))\right]^+.\end{aligned} \tag{4.11}$$

In the above formula, the $x_i^k(t)$ and $c_{i,j}^k(t)$ are the maximizers of $D_1(\mu)$ and $D_2(\mu)$, given $\mu(t)$. $h(t)$ is a positive scalar stepsize. Finally, '+' denotes the projection onto the set $\mathbf{R}_+$ of non-negative real numbers.

4.5 Distributed Implementation

In this section, we describe how the proposed QoI based data selection and transmission scheme can be implemented in a distributed and scalable way. As aforementioned in Section 4.3, every v time slots the events in the sliding window are updated and the source nodes' clustering results of the newly-occurred events are sent to the sink of each mission, where the reliability and redundancy estimates of the sensor nodes are updated. The QoI scores are then sent back to the source nodes. Since both the clustering results and the QoI estimates are numeric data, little communication overhead is incurred during this process.

Upon receiving the reliability score, each source node can derive x_i^k based on Eqn. (4.9) for each of the virtual sources connected to it, given the shadow price of the current slot μ_i^k. Subsequently, the neighboring nodes exchange their μ_i^k, and solve the routing and scheduling problem, namely, decide which source (relay) nodes will have chance to sense (transmit) in the next slot, through the strategies as we discussed previously in Section 4.4.2. Once the work for the current time slot is done, each node updates its shadow price according to Eqn. (4.11).

4.6 Performance Evaluation

In this section, we evaluate our proposed schemes. Results on both synthetic data and recorded audio data are presented and discussed.

4.6.1 Experimental Settings

Network Topology: We consider a randomly generated sensor network consisting of 50 sensor nodes. The network topology is shown in Fig. 4.5. In this network, two missions are being undertaken, and each of them involves 10 source nodes (the nodes with index numbers) and a single sink (the hexagon node). The sensor nodes for the two missions can be distinguished by their colors (red for mission 1 and blue for mission 2). In this experiment, we randomly assign the link capacities and data collection rates of the source nodes. We use synthetic data for mission 1 and real audio data for mission 2.

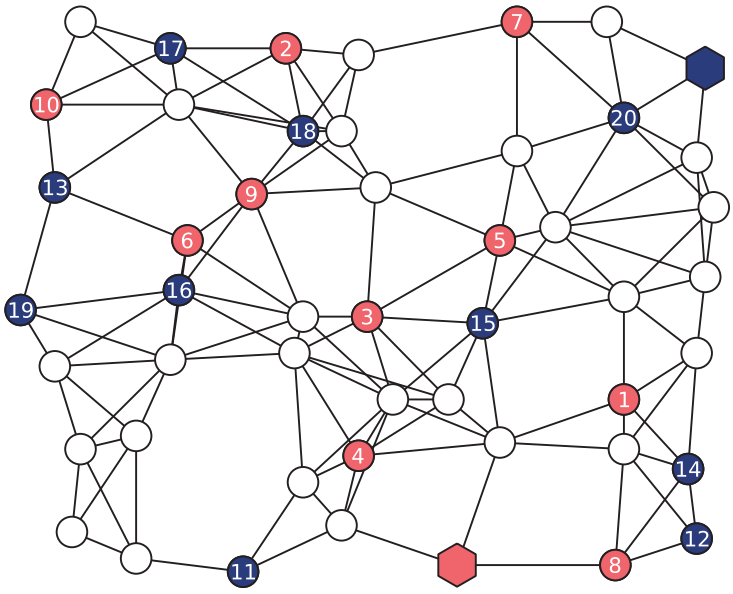


Figure 4.5: The sensor network used in the experiment.

Synthetic Data: Suppose there are 10 different types of sensors, corresponding to 10 features of the events (e.g., temperature, humidity, etc). We randomly generate events from a Gaussian mixture model with 5 components, each of which corresponds to a class. For the first 7 of 10 source nodes, we randomly assign a subset of the previously defined 10 types of sensors to each of them,

and add random Gaussian noise to each type of sensor assigned to this node. We assume that the 8th node is collecting data completely irrelevant to the mission, and thus generate its data from a different distribution. The last two nodes are the duplicates of the first two nodes, with some additional noise added to their data. Therefore, node 1 and 9, node 2 and 10 are two pairs of redundant nodes.

Audio Data: The audio clips we use in this experiment include the sounds of tank, helicopter, and machine gun, corresponding to 3 different classes. We cut the audio clips into pieces with equal time duration, and make a copy for each node. Similar to the synthetic data, we add random noise to the records of the first 7 source nodes with various SNRs. The 8th node is supposed to record fundamentally irrelevant sounds such as crowd talking. The last two nodes are made redundant to node 11 and 12 correspondingly. In the experiment, we extract the MFCC (Mel-Frequency Cepstral Coefficients) features from each audio piece, and feed them as the input to the clustering algorithm.

4.6.2 Experimental Results

We evaluate the proposed schemes using the aforementioned data and experimental settings. In this experiment, we set the width w and the stepsize v of the sliding window to be 1000 and 500 time slots. The experiment spans 12500 time slots, and thus the window slides for 25 times.

Table 4.5: Noise Level

Node	1	2	3	4	5	6	7	8	9	10
Noise (STD)	50	30	1	10	40	5	60	∞	53	31
Node	11	12	13	14	15	16	17	18	19	20
Noise (SNR)	-1	20	10	0	-2	10	5	∞	-1	19

Groundtruth QoI: Table 4.5 lists the level of the noise added to each source node. The metric of noise level is the standard deviation (STD) (given Gaussian noise) for mission 1 and signal-to-noise ratio (SNR) for mission 2. For the irrelevant nodes (node 8 and 18), we regard their noise level as infinity. The noise levels of the duplicate nodes (node 9, 10, 19, and 20) are derived through

cumulating the noise of the original nodes from which their data are copied and the additional noise injected later. The noise level can be regarded as the groundtruth reliability of the sensor nodes. The groundtruth redundancy, as mentioned in Section 4.6.1, is as follows: In either mission we set the node pair 1&9 and 2&10 to be redundant, and the other node pairs are irredundant.

Next, we use the groundtruth QoI to evaluate the proposed QoI metrics.

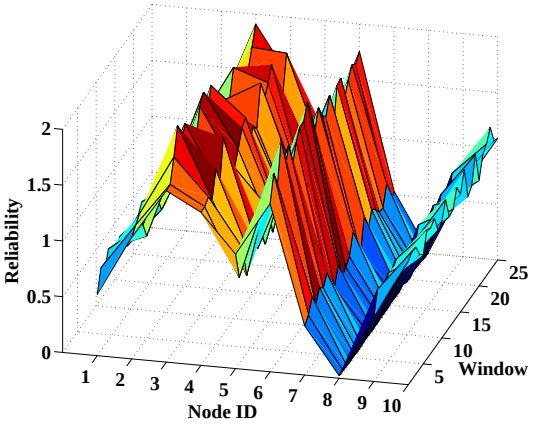


Figure 4.6: Node-window reliability score on synthetic data

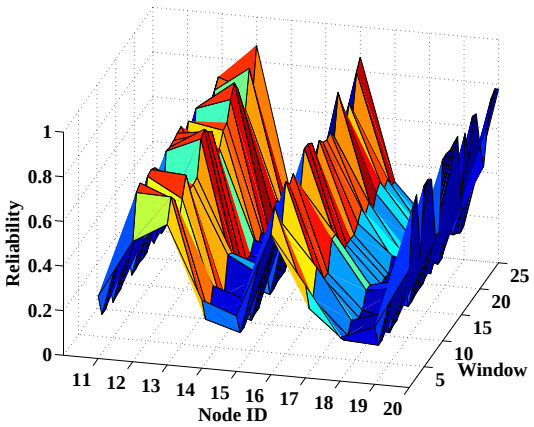


Figure 4.7: Node-window reliability score on sound data

Reliability Estimation: We first estimate the data reliability of sensor nodes according to the proposed metric. Figure 4.6 and 4.7 show the reliability scores of the source nodes for all the windows in mission 1 and 2. As can be seen, our measures match perfectly with the groundtruth reliability scores, i.e., the levels of noise added to the sensor nodes. For example, node 3 has a noise level of 1 (STD), meaning it is able to capture data with virtually no noise, reflected as the high (peak) reliability measure of node 3 shown in Fig.4.6. As another example, node 15 has a noise level of -2 (SNR), which means the intensity of the noise added is greater than the original sound captured by the node. Therefore, node 15 would be rather unreliable, reflected as the low reliability measure (valley) at node 15 in Fig.4.7. Figure 4.8 and 4.9 show the per-node reliability traces through the 25 data windows for both missions. As seen, all nodes generally maintain their reliability measures for all the windows. The fluctuations are due to the added noise and the intrinsic nature of the sound data (certain segments of the sounds are indistinguishable).

Redundancy Estimation: Figure 4.10 and 4.11 plot the redundancy measures of the sensor

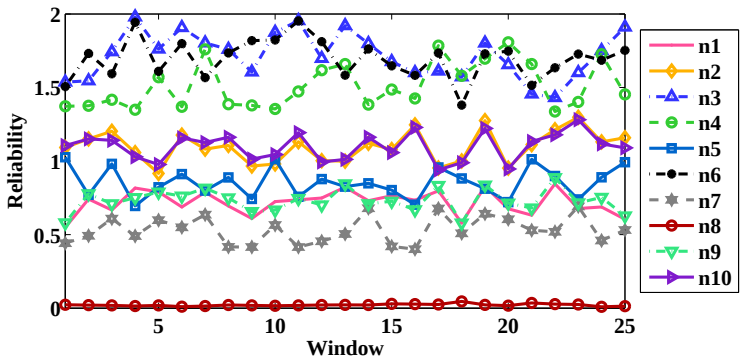


Figure 4.8: Node reliability evolution on synthetic data

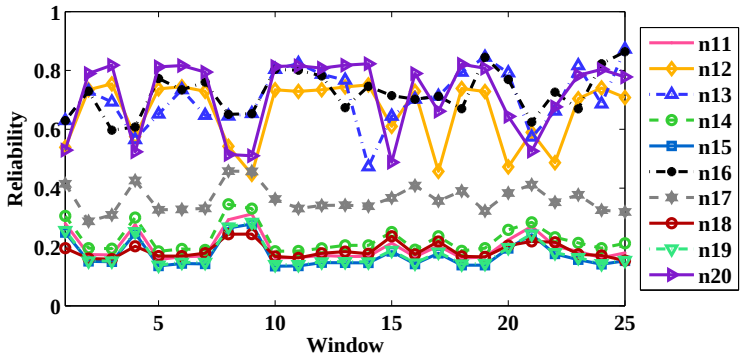


Figure 4.9: Node reliability evolution on sound data

nodes. In particular, we use Jaccard coefficient for synthetic data, and Rand statistic for audio data. In either figure, the measures for two pairs of redundant nodes (node 1&9 and node 2&10) and two pairs of irredundant nodes (node 1&4 and node 2&5) are displayed. As can be seen, there is a clear gap between the measures of redundant and irredundant nodes.

Convergence: Figure 4.12 shows the evolution of the objective value (Eqn. (4.7)), i.e., the aggregate reliability over a period of 1500 time slots or 3 windows. As one can see, within each time window, it converges quickly according to the updated QoI measures and oscillates around the optimal values. This oscillating behavior can be interpreted as due to the process of link scheduling.

Source Selection: Figure 4.13 illustrates the selection process of two redundant nodes. In this figure, the vertical axis represents the selection result, where 1 means the corresponding node is selected to collect data for the current time slot and 0 means it is not selected. As can be seen, the two redundant nodes are never selected at the same time.

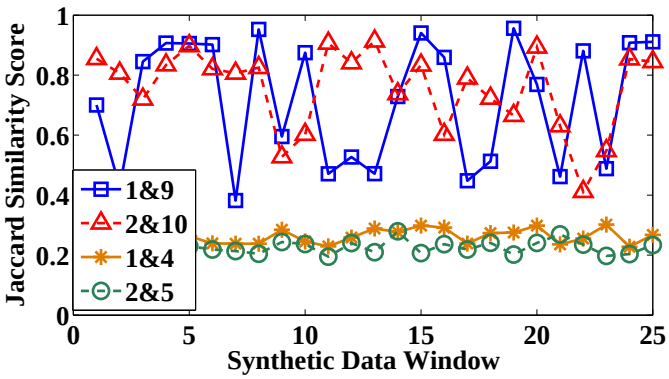


Figure 4.10: Jaccard coefficient of synthetic data

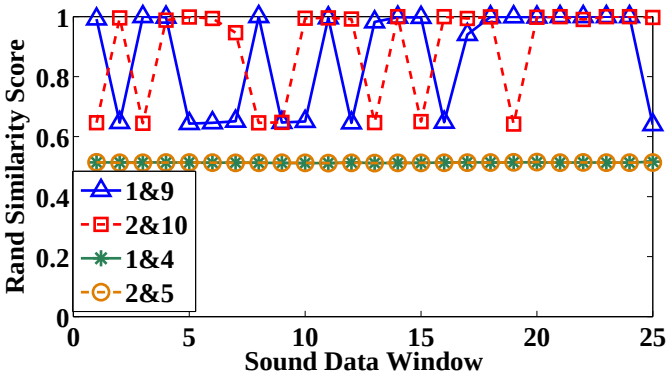


Figure 4.11: Rand statistic of audio data

Figure 4.14 and 4.15 demonstrate the data selection of all the source nodes during the total 25 time window. In the two figures, the darkness of each small rectangle represents the frequency at which that particular node is selected in the corresponding time window. Pure black means the node is selected all the time, while pure white means the node is never selected in any of the time windows.

Table 4.6 lists the average selection rate of each source of all the windows (the row named optimal selection). At the first glance, the result does not strictly follow the groundtruth QoI. For example, node 1 keeps working almost all the time, while node 3 has little chance to be selected, although its noise level is much smaller than that of node 1. The problem lies in the network resource constraint. If a node has very little available resource along the path to the sink, it will not be selected frequently even if its data is highly reliable. For comparison, we design a naive baseline scheme in which all the source nodes are treated equally. Specifically, we set the reliability

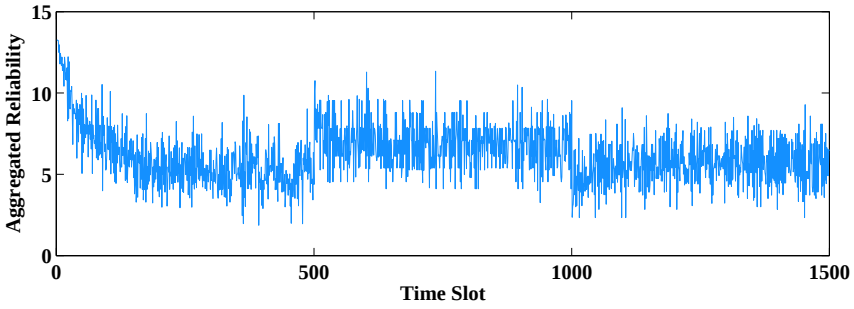


Figure 4.12: The convergence of the aggregate reliability.

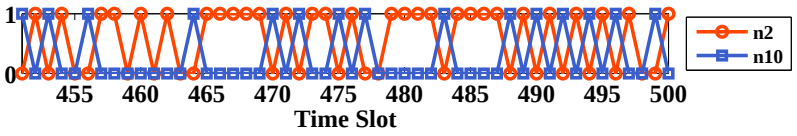


Figure 4.13: Selection of redundant nodes

values of all the sources, i.e., p_i^k in the objective function of $\mathbf{P}$, to be equal. The source selection rates of the baseline scheme are listed in Table 4.6 (the row named uniform selection). As the numbers suggest, compared with uniform selection, the proposed optimal selection scheme is more compatible with the groundtruth QoI. For example, node 6 has a rather low noise level, thus our optimal selection scheme picks it nearly half of the time; on the other hand, the uniform selection scheme would only select it one out of ten times. As another example, node 16 has the second highest SNR among all the nodes in mission 2. In our optimal selection scheme, the chance that this node is picked is almost two out of three; but the uniform scheme only selects this node less than 20% of the time.

Table 4.6: Source Selection

Node	1	2	3	4	5	6	7	8	9	10
Optimal Selection	.99	.57	.09	.33	.31	.43	.6	0	0	.11
Uniform Selection	1	.3	.11	.2	.39	.1	.99	.25	.06	.1
Node	11	12	13	14	15	16	17	18	19	20
Optimal Selection	.01	.39	1	.1	.99	.65	.08	.03	.01	.97
Uniform Selection	.12	.15	1	.53	1	.19	.12	.28	.14	.66

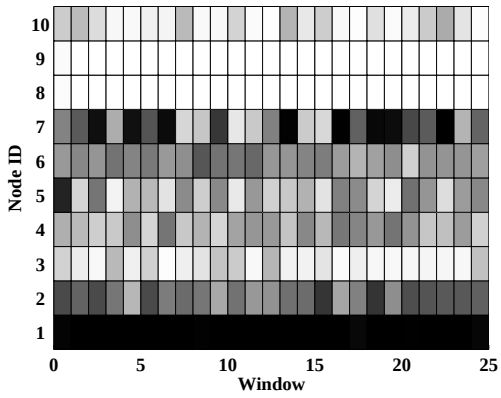


Figure 4.14: Source Selection of Synthetic Data

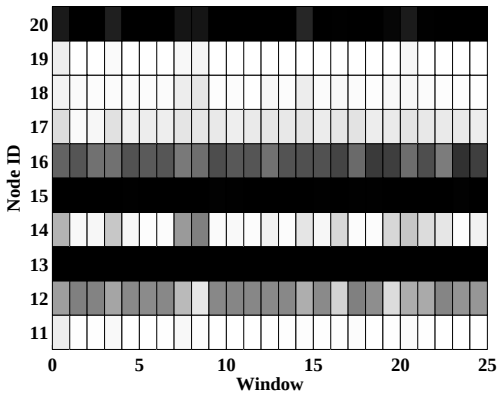


Figure 4.15: Source Selection of Audio Data

As a final evaluation measure, we compute the window-averaged total reliability scores of the two selection schemes for both missions. For each window of each node in each mission, we take the product of the node reliability score and its selection rate, we then average the products over all the windows, and sum the averages across all the nodes. For the first mission, the uniform selection scheme has a score of 2.7214; our optimal selection scheme achieves a score of 3.5345, an 29.88% advantage. For the second mission, the two scores are 1.8513 and 2.3933, with our optimal scheme claiming a 29.28% advantage.

4.7 Related Work

Sometimes, data abstraction and compression techniques such as dimensionality reduction [10, 25, 40] and compressed sensing [20, 58, 85] can be employed to reduce the size of sensory data reported by the sensor nodes. First, in this work we focus on the scenarios where the raw data are desired to be delivered as is. Second, in the scenarios where data abstraction and compression are allowed, our proposed data selection and transmission scheme is still useful when the system resources are not enough to deliver the abstracted or compressed data.

The distributed joint design of data selection and transmission proposed in this chapter borrow the idea of network utility maximization (NUM), which has been widely studied in the context of both wired networks [18, 43, 46, 47, 56] and wireless networks [12, 23, 53, 54]. Different from existing NUM work that aim at optimizing system parameters (e.g., throughput, delay), we take

into consideration the quality of transmitted data (i.e., data reliability and redundancy), and thus can achieve more information gain from the application perspective.

4.8 Conclusions

In this chapter, we identify the two aspects of QoI, data reliability and redundancy, for decision making tasks in cyber-physical systems, and propose metrics to estimate them. Then, we develop a data selection and transmission service which can maximize the reliability of the delivered data, with data redundancy being removed. The key of the solution lies in the decomposition of the data selection and transmission problem. A distributed algorithm is designed to solve the decomposed problem separately.

Chapter 5

Generalized Decision Aggregation

5.1 Introduction

The aforementioned decision aggregation approaches, though yielding reasonably good performance in certain cases, suffer from a major limitation. That is, they only take as input discrete decision information. Sometimes individual sensors may not be quite confident about their decisions due to various reasons, such as incomplete or noisy observations. In this case, if each sensor's confidence information (the probability that it "believes" the observed event belongs to each candidate class) can also be taken into consideration, we should be able to further improve the final decision accuracy. For example, suppose a vehicle is observed by three sensors that try to determine whether it is a tank or a truck (assuming it is actually a tank). Each sensor then provides a confidence probability vector corresponding to its belief in the vehicle being a tank or a truck. Suppose the three probability vectors are (0.99, 0.01), (0.49, 0.51), and (0.49, 0.51). In this case, only one sensor predicts the vehicle to be a tank, however, deciding so with high confidence, as opposed to the other two that both vote for a truck but with confidences not far from that of random guess, as reflected by their decision probability vectors. Therefore, we expect a reasonable decision aggregation scheme to output tank as the aggregated decision, as opposed to only taking discrete decision labels as input, which would likely get the wrong answer, favoring the majority though incorrect decision in our example.

In practice, the decision probability vectors can be obtained from both traditional hardware-centric cyber-physical systems where sensor nodes conduct explicit classification computations, and newly-emerged people-centric sensing paradigms where people conduct implicit classifications through their logical reasoning. For example, when working with hardware sensor nodes, we can adopt classification algorithms that derive decision probabilities through heuristic metrics like

the distance between the observed data and the decision boundary learned from training data; When having people carry out sensing tasks, we can ask each participant to explicitly provide the confidence level of each decision made.

The goal of this chapter is to develop a *generalized decision aggregation* (GDA) framework for cyber-physical systems that can address the above challenge, by taking as input each individual sensor's decision probability vectors and computing the aggregated decision (class label) for all events under observation. In pursuing the generalizability such that it is applicable to a full range of sensing scenarios, our proposed GDA framework bears the following properties.

i) Each individual sensor's reliability level is explicitly accounted for when GDA integrates individual decisions. A sensor's reliability information is important as it reflects the general quality of information it can provide. The aggregated decision should favor more reliable sensors and weigh less unreliable ones instead of treating all individuals equally. In reality, however, the reliability information is usually unknown a priori. To address this, in our GDA framework, the sensors' reliability is estimated along with the decision aggregation process and provided as part of the final outputs to the user.

ii) Different from traditional decision aggregation schemes that assume all the events are observed by all the sensors, the proposed GDA framework is able to handle the scenarios where different sensor nodes observe different sets of events whose numbers of possible classes may also be different. Doing so enables us to seize more opportunities to estimate sensor reliability, thus leading to better final decision accuracy.

iii) In order to be applied to newly emerged sensing scenarios where people are playing increasingly more critical roles, which implies more opportunities for ground truth label collection, we design our GDA framework from ground up to be able to cope with any availability level of label information, and do so in a dynamic and intelligent manner.

In summary, our GDA framework addresses challenges in carrying out decision making tasks in cyber-physical systems by more naturally and organically modeling the sensing and decision making processes in dynamic and intelligent manners, it thus can be applied to a full spectrum of distributed sensing domains.

The rest of the chapter is organized as follows. Section 5.2 provides an overview of the system model and architecture. In Section 5.3, we formulate the generalized decision aggregation problem as an optimization program. An efficient algorithm is presented in Section 5.4 to solve this problem. The proposed algorithm is evaluated in Section 5.5. Then, we summarize the related work in Section 5.6, and conclude the chapter in Section 5.7.

5.2 System Overview

We now give an overview of our system model and architecture.

5.2.1 System Model

We consider a sensing system consisting of n sensor nodes that are denoted by $\mathcal{S} = \{s_i,\ i = 1, 2, \ldots, n\}$. The sensor nodes collect information about the events taking place within their sensing ranges, and classify these events into predefined classes. Formally, we let $\mathcal{E} = \{e_i | i = 1, 2, \ldots, t\}$ denote the sequence of events (sorted in chronological order) observed by the sensor nodes. Generally, each sensor observes a subset of events, and each event is observed by a subset of sensors. The relationship between sensor nodes and events can be represented as a bipartite graph, called *belief graph*, where vertices are partitioned into sensors and events, and edges represent the observation relationships of sensor-event pairs, as illustrated in Fig. 5.1.

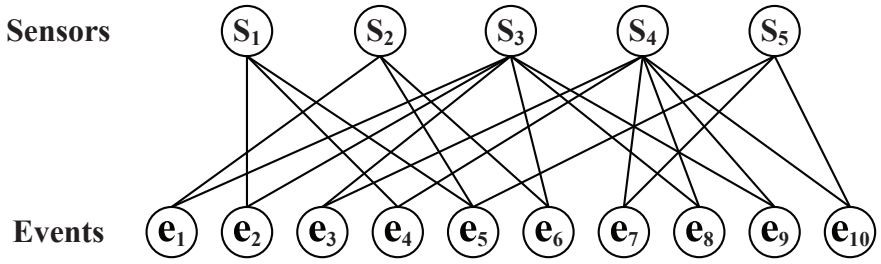


Figure 5.1: An example of belief graph

In this case, suppose the mission of the sensing system is to classify different vehicle types, specifically, to find out whether an observed vehicle is a tank, a jeep, or a truck. As shown, 10 events are observed by 5 sensor nodes. Each event corresponds to a vehicle. Each sensor can be either a sensing device deployed on the roadside, or a pedestrian in the vicinity.

5.2.2 System Architecture

In this section, we provide an overview of the system architecture. The system contains three modules: a data classification module, a decision aggregation module, and a feedback module. They are deployed on two different platforms: sensor nodes, and the base station. Figure 5.2 illustrates the system architecture. We next discuss each of these three modules in more detail.

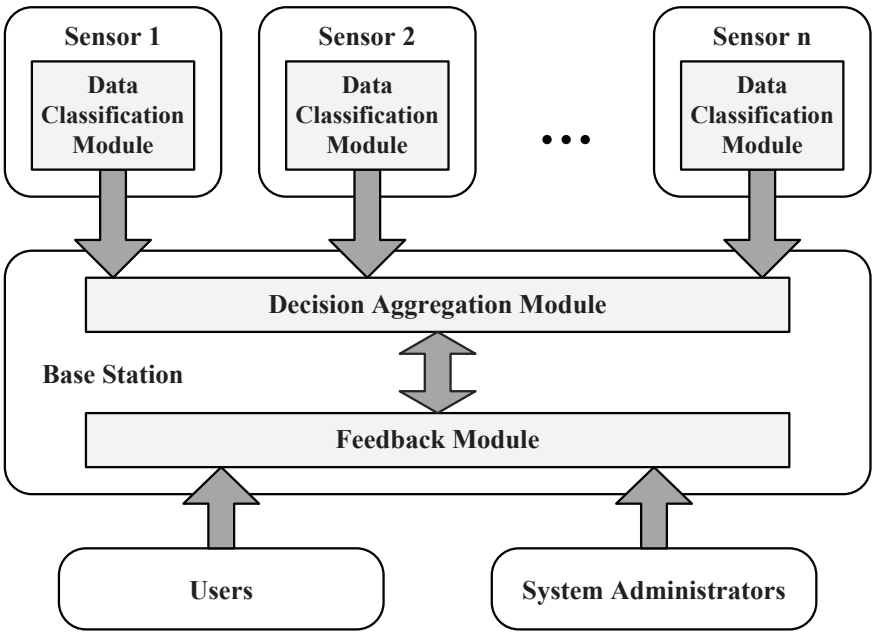


Figure 5.2: System Architecture

Data Classification Module

The data classification module runs on individual sensor nodes. It performs four major tasks as listed below:

- Data Collection: Access various on-board sensors to monitor events.
- Data Processing: Extract features from the raw data that can capture the characteristics of different events.
- Data Classification: Locally classify the events observed by the sensor nodes based on the features extracted from the raw data.

- Decision Delivery: Upload the classification result (i.e., decision vector) of each sensor node to the base station.

Decision Aggregation Module

As shown in Fig. 5.2, the decision aggregation module resides on the base station. Its role is twofold:

- Decision Collection: Listen for incoming connections and receive decision vectors from individual sensor nodes. Decision vectors can be either directly piped to the aggregation engine, or stored in database for later batch processing.
- Decision Aggregation: Combine decision vectors from multiple sensor nodes, and predict the class label of each unlabeled event.

Feedback Module

The feedback module is included to visually present data collection and/or decision results via a web service interface. It can also be used by system administrators and users to correct decision errors or provide ground truth information, which is then sent back to individual sensors for dynamic and adaptive performance improvement.

5.3 Problem Formulation

With the previously defined notations and terminologies, we formulate the generalized decision aggregation problem as an optimization program on the belief graph. In this section, we first introduce the variables and constants involved in the optimization program, then give detailed descriptions on the objective function as well as the constraints.

5.3.1 Variables and Constants

Constants: The constants of the optimization program are the inputs to the decision aggregation module, including the following:

- Belief Graph: We summarize the previously defined belief graph into an affinity matrix called observation matrix $\mathbf{A} = (a_{ij})_{t \times n}$, where a_{ij} indicates whether event e_i is observed by sensor s_j.

- Individual Decision: The decision of node s_j for e_i is a probability vector denoted by $\mathbf{d}_i^j = (d_{i1}^j, \ldots, d_{im_i}^j)$, where m_i is the number of possible classes of event e_i. In this vector, each element probability, say d_{ik}^j, represents the confidence level in which s_j "believes" the observed event belongs to the k-th class. For example, suppose sensor s_1 in Fig. 5.1 outputs a decision vector for event e_2 (i.e., $\mathbf{d}_2^1 = (0.8, 0.1, 0.1)$). This implies that s_1 believes that with 80% probability e_2 is a tank, and is a jeep or truck with 10% probability each.

Variables: The variables of the optimization program serve as the outputs of the decision aggregation module, including the following:

- Aggregated Decision: The aggregated decision for an event e_i is also a probability vector, denoted by $\mathbf{x}_i = (x_{i1}, \ldots, x_{im_i})$. It represents the consensus of the sensor nodes on the probability that e_i belongs to each candidate class.

- Sensor Reliability: As discussed in Section 5.1, the reliability levels of individual sensor nodes should be taken into account when aggregating the decisions of multiple nodes. To capture sensor reliability, we associate each node, say s_j, with a non-negative weight w_j, where higher weights indicate higher reliability[1].

[1]In our abstract mathematical formulation, the reliability of a sensor node is modeled as a single weight variable, reflecting the degree to which the data from this node contributes to the targeted decision making task, automatically derived from analyzing the sensory data. This highly generalized reliability model thus enables our mathematical formulation to be easily applicable to a wide range of cyber-physical systems consisting of heterogeneous sensors with various intrinsic reliability definitions, some of which can potentially get quite complex in themselves—for example, in multimedia systems, video sensor reliability (and the corresponding video data quality) usually refers to parameters like frame resolution, number of pixels per frame, bits per pixel, etc. It is, therefore, difficult, if not impossible, to come up with a concise formulation that is both generalizable and capable of capturing the complex specifics of heterogeneous sensor nodes. Since it is our intention to construct an abstract mathematical formulation that is generalizable to a wide range of cyber-physical applications, we do not attempt to capture the individual reliability models of any specific sensors into our generalized formulation.

5.3.2 Optimization Program

Given the constants and variables defined above, we formulate the following optimization program:

$$\mathbf{P}: \min \quad \sum_{i=1}^{t}\sum_{j=1}^{n} a_{ij} w_j ||\mathbf{x}_i - \mathbf{d}_i^j||^2 \tag{5.1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} \exp(-w_j) = 1 \tag{5.2}$$

$$\mathbf{x}_i \geq \mathbf{0}, \quad |\mathbf{x}_i| = 1 \quad \text{for} \quad i = 1, 2, \ldots, t \tag{5.3}$$

Objective Function: The objective function (5.1) aims at minimizing the disagreement over the belief graph, namely, the weighted summation of the distances between the decisions of individual sensor nodes and the aggregated decision. In this case, we use squared L2 norm as the distance function:

$$||\mathbf{x}_i - \mathbf{d}_i^j||^2 = (x_{i1} - d_{i1}^j)^2 + (x_{i2} - d_{i2}^j)^2 + \ldots + (x_{im_i} - d_{im_i}^j)^2$$

Intuitively, the optimal aggregated decision should be close to the majority of individual decisions[2]. Furthermore, the sensors with higher reliability score (i.e., w_j) should have more impact on the weighted summation. In other words, more reliable sensors would incur higher penalties if they deviate far away from the aggregated decision, as compared to less reliable ones. This way, the objective function tends to be minimized when the aggregated decision agrees with that of reliable sensors.

Constraints: Next, we elaborate on the constraints that our objective function is subject to.

- Reliability Constraint (5.2) is a regularization function. It is used to prevent the sensor weight w_j from going to infinity, otherwise the optimization problem would become unbounded. In fact, the most straightforward choice of regularization function could be $\sum_{j=1}^{n} w_j = 1$, which is unsuitable for our purpose as an optimal solution is achieved when the aggregated decision is set to that of any single sensor, whose weight is set to 1 and the rest sensors 0. Therefore we propose to formulate the regularization function using the sum over exponential value of weights. Exponential function is used to regularize weights so that they are rescaled

[2]Here we assume the majority of the sensor nodes are functioning appropriately and thus can make reasonable decisions.

by logarithm (the range of weights becomes smaller). One advantage of this regularization formulation is that the problem becomes convex under this regularization function which leads to a closed-form optimal solution.

- Decision Constraint (5.3) is used to guarantee that the elements of the decision probability vector $\mathbf{x}_i$ be nonnegative, and sum to 1.

Unfortunately, **P** is not a convex program, as shown in Theorem 12. This makes it difficult to find the global optimal solution.

Theorem 12. **P** is not a convex program.

Proof. To show that **P** is not a convex program, we just need to find a counter-example, namely, an objective function of **P** that does not satisfy the definition of convex function.

Consider a simple scenario: there is only one event that is observed by two sensors. In this case, the objective function becomes:

$$f(w_1, w_2, \mathbf{x}) = w_1||\mathbf{x} - \mathbf{d}^1||^2 + w_2||\mathbf{x} - \mathbf{d}^2||^2$$

Denote a solution of **P** by $\mathbf{u} = (w_1, w_2, \mathbf{x})^{\mathrm{T}}$. Now, we choose two feasible solutions $\mathbf{u}_1 = (1, 0, \mathbf{d}^1)^{\mathrm{T}}$, $\mathbf{u}_2 = (0, 1, \mathbf{d}^2)^{\mathrm{T}}$. Clearly, $f(\mathbf{u}_1) = f(\mathbf{u}_2) = 0$.

Let $0 < \alpha < 1$, then we have

$$\begin{aligned} f(\mathbf{u}) &= f(\alpha\mathbf{u}_1 + (1-\alpha)\mathbf{u}_2) \\ &= f(\alpha(1, 0, \mathbf{d}^1)^{\mathrm{T}} + (1-\alpha)(0, 1, \mathbf{d}^2)^{\mathrm{T}}) \\ &= f(\alpha, 1-\alpha, \alpha\mathbf{d}^1 + (1-\alpha)\mathbf{d}^2) \\ &= \alpha||(\alpha-1)\mathbf{d}^1 - (\alpha-1)\mathbf{d}^2||^2 + (1-\alpha)||\alpha\mathbf{d}^1 - \alpha\mathbf{d}^2||^2. \end{aligned}$$

It is clearly seen that as long as we have the inequality $\mathbf{d}^1 \neq \mathbf{d}^2$, the following relation will hold:

$$f(\alpha\mathbf{u}_1 + (1-\alpha)\mathbf{u}_2) > 0 = \alpha f(\mathbf{u}_1) + (1-\alpha)f(\mathbf{u}_2).$$

This contradicts with the definition of convex function, so **P** is not a convex program. □

Next, we present an efficient approximate solution for the optimization program $\mathbf{P}$.

5.4 Generalized Decision Aggregation

We propose to solve $\mathbf{P}$ using the block coordinate descent method [5]. The basic idea is as follows: In each iteration, we update the values of sensor reliability and aggregated decision alternatively and separately. In particular, in the first step, we fix the weight (w_j) of each sensor node, and solve $\mathbf{P}$ with respect to the aggregated decision ($\mathbf{x}_i$) only. In the second step, $\mathbf{x}_i$ is fixed and $\mathbf{P}$ is solved with respect to w_j. The two-step process is repeated until convergence, which is guaranteed by the property of the block coordinate descent method. That is, if we can find the optimal aggregated decision (sensor reliability) when sensor reliability (aggregated decision) is fixed, convergence can be achieved [5]. Next, we give detailed explanation on these two steps, and show that each step itself is convex, and thus has a globally optimal solution.

5.4.1 Updating Aggregated Decision

When the reliability w_j of each sensor is fixed, we update the aggregated decision $\mathbf{x}_i$ for each event in order to minimize the weighted distances between $\mathbf{x}_i$ and the decisions $\mathbf{d}_i^j$ made by individual sensor nodes:

$$\mathbf{P_x}: \min \quad \sum_{i=1}^{t}\sum_{j=1}^{n} a_{ij} w_j ||\mathbf{x}_i - \mathbf{d}_i^j||^2$$
$$\text{s.t.} \quad \mathbf{x}_i \geq \mathbf{0}, \quad |\mathbf{x}_i| = 1 \quad \text{for} \;\; i = 1, 2, \ldots, t$$

Different from $\mathbf{P}$, $\mathbf{P_x}$ has only one set of variables, which are the $\mathbf{x}_i$'s. Note that w_j's are just constants in $\mathbf{P_x}$, whose convexity is shown in Theorem 13.

Theorem 13. $\mathbf{P_x}$ is a convex program.

Proof. Since the constraints of $\mathbf{P_x}$ are all linear, we only need to show the objective function of $\mathbf{P_x}$ is convex.

$$f(\mathbf{x}) = \sum_{i=1}^{t} \Big\{ \sum_{j=1}^{n} a_{ij} w_j ||\mathbf{x}_i - \mathbf{d}_i^j||^2 \Big\}$$

For each event e_i,

$$\begin{aligned} f(\mathbf{x}_i) &= \sum_{j=1}^{n} a_{ij} w_j ||\mathbf{x}_i - \mathbf{d}_i^j||^2 \\ &= \sum_{j=1}^{n} a_{ij} w_j (\mathbf{x}_i - \mathbf{d}_i^j)^{\mathrm{T}} (\mathbf{x}_i - \mathbf{d}_i^j) \\ &= \sum_{j=1}^{n} a_{ij} w_j (\mathbf{x}_i{}^{\mathrm{T}} \mathbf{x}_i - 2\mathbf{x}_i{}^{\mathrm{T}} \mathbf{d}_i^j + \mathbf{d}_i^{j\mathrm{T}} \mathbf{d}_i^j) \\ &= \mathbf{x}_i{}^{\mathrm{T}} [(\sum_{j=1}^{n} a_{ij} w_j)\mathbf{I}] \mathbf{x}_i - 2 \sum_{j=1}^{n} a_{ij} w_j \mathbf{x}_i{}^{\mathrm{T}} \mathbf{d}_i^j + \sum_{j=1}^{n} a_{ij} w_j \mathbf{d}_i^{j\mathrm{T}} \mathbf{d}_i^j \end{aligned}$$

Since each event is observed by at least one sensor, there is at least one a_{ij} equal to 1, and w_j is always positive, we can infer that $\sum_{j=1}^{n} a_{ij} w_j > 0$. Thus, the Hessian matrix $(\sum_{j=1}^{n} a_{ij} w_j)\mathbf{I}$ is strictly positive definite ($\mathbf{I}$ is the identity matrix). Therefore, $f(\mathbf{x})$ is a convex function and $\mathbf{P}_{\mathbf{x}}$ is a convex program. □

This ensures that we can find globally optimal aggregated decisions. The detailed steps are as follows. First, we denote the objective function by

$$f(\mathbf{x}) = \sum_{i=1}^{t} \sum_{j=1}^{n} a_{ij} w_j \sum_{k=1}^{m_i} (x_{ik} - d_{ik}^j)^2,$$

then the optimal solution can be obtained through setting the partial derivative with respect to $\mathbf{x}$ to zero,

$$\frac{\partial f(\mathbf{x})}{\partial x_{ik}} = \sum_{j=1}^{n} 2 a_{ij} w_j (x_{ik} - d_{ik}^j) = 0$$

for $i = 1, 2, \ldots, t$ and $k = 1, 2, \ldots, m_i$.

Solving this equation, we are able to get the optimal value of x_{ik}:

$$x_{ik} = \frac{\sum_{j=1}^{n} a_{ij} w_j d_{ik}^j}{\sum_{j=1}^{n} a_{ij} w_j}.$$

Therefore, the optimal aggregated decision vector is actually the weighted average of individual decision vectors:

$$\mathbf{x}_i = \frac{\sum_{j=1}^{n} a_{ij} w_j \mathbf{d}_i^j}{\sum_{j=1}^{n} a_{ij} w_j}. \tag{5.4}$$

One should note that when we solve for $\mathbf{x}_i$, we do not take into account the decision constraint (Eqn. (5.3)). This is because the aggregated decisions obtained from Eqn. (5.4) can automatically satisfy the constraint, as shown in Theorem 14.

Theorem 14. The aggregated decisions obtained from Eqn. (5.4) automatically satisfy the decision constraint.

Proof. Since each individual decision $\mathbf{d}_i^j$ is a probability vector, obviously we have $\mathbf{d}_i^j \geq 0$ and $|\mathbf{d}_i^j| = 1$. Thus, it can be derived that

$$\begin{aligned}
|\mathbf{x}_i| &= \sum_{k=1}^{m_i} x_{ik} = \sum_{k=1}^{m_i} \frac{\sum_{j=1}^{n} a_{ij} w_j d_{ik}^j}{\sum_{j=1}^{n} a_{ij} w_j} \\
&= \frac{\sum_{j=1}^{n} a_{ij} w_j \sum_{k=1}^{m_i} d_{ik}^j}{\sum_{j=1}^{n} a_{ij} w_j} \\
&= \frac{\sum_{j=1}^{n} a_{ij} w_j |\mathbf{d}_i^j|}{\sum_{j=1}^{n} a_{ij} w_j} \\
&= \frac{\sum_{j=1}^{n} a_{ij} w_j}{\sum_{j=1}^{n} a_{ij} w_j} = 1
\end{aligned}$$

Moreover, since both a_{ij} and w_j are nonnegative, it is clear that $x_{ik} \geq 0$. Therefore, $\mathbf{x}_i$'s automatically satisfy the decision constraint. □

5.4.2 Updating Sensor Reliability

Next, we fix the values of the aggregated decision $\mathbf{x}_i$, and update the reliability of each sensor w_j through solving the following optimization program:

$$\begin{aligned}
\mathbf{P_w}: \min \quad & \sum_{i=1}^{t} \sum_{j=1}^{n} a_{ij} w_j ||\mathbf{x}_i - \mathbf{d}_i^j||^2 \\
\text{s.t.} \quad & \sum_{j=1}^{n} \exp(-w_j) = 1
\end{aligned}$$

Similar to the previous step, $\mathbf{P_w}$ has only one set of variables, the w_j's. And the decision constraint (Eqn. (5.3)) in $\mathbf{P}$ is just constant here. $\mathbf{P_w}$ is clearly convex since the objective function is linear with respect to w_j, while the constraint is a convex function.

We use the method of Lagrange multipliers to solve $\mathbf{P_w}$. We first take a look at the Lagrangian of $\mathbf{P_w}$:

$$L(\mathbf{w}, \lambda) = \sum_{i=1}^{t}\sum_{j=1}^{n} a_{ij} w_j ||\mathbf{x}_i - \mathbf{d}_i^j||^2 + \lambda(\sum_{j=1}^{n} \exp(-w_j) - 1)$$

In $L(\mathbf{w}, \lambda)$, λ is a Lagrange multiplier, corresponding to the reliability constraint. It can be interpreted as the "shadow price" charged for the violation of the constraint.

Let the partial derivative of Lagrangian with respect to w_j be 0:

$$\frac{\partial L(\mathbf{w}, \lambda)}{\partial w_j} = \sum_{i=1}^{t} a_{ij} ||\mathbf{x}_i - \mathbf{d}_i^j||^2 - \lambda \exp(-w_j) = 0,$$

we can get

$$\frac{\sum_{i=1}^{t} a_{ij} ||\mathbf{x}_i - \mathbf{d}_i^j||^2}{\lambda} = \exp(-w_j). \tag{5.5}$$

Summing both sides over j's, we have,

$$\frac{\sum_{j=1}^{n}\sum_{i=1}^{t} a_{ij} ||\mathbf{x}_i - \mathbf{d}_i^j||^2}{\lambda} = \sum_{j=1}^{n} \exp(-w_j) = 1,$$

from which we can derive that

$$\lambda = \sum_{j=1}^{n}\sum_{i=1}^{t} a_{ij} ||\mathbf{x}_i - \mathbf{d}_i^j||^2. \tag{5.6}$$

Plugging Eqn. (5.6) into Eqn. (5.5), we obtain a closed-form solution of reliability:

$$w_j = \log\Big(\frac{\sum_{j=1}^{n}\sum_{i=1}^{t} a_{ij} ||\mathbf{x}_i - \mathbf{d}_i^j||^2}{\sum_{i=1}^{t} a_{ij} ||\mathbf{x}_i - \mathbf{d}_i^j||^2}\Big). \tag{5.7}$$

5.4.3 Algorithm

The detailed steps of the generalized decision aggregation (GDA) algorithm are shown in Algorithm 5. The algorithm takes as input the observation matrix $\mathbf{A}$ as well as the individual decision of each sensor s_j for each event e_i (i.e., $\mathbf{d}_i^j$). It starts by initializing the aggregated decisions randomly (line 1). The iterative process then begins in line 3. First, we collect the aggregated decision of each event observed by a sensor s_i, and update its reliability via Eqn. (5.7) (line 5). Then, the

sensors' reliability information are used to consolidate the aggregated decision of each event e_i through Eqn. (5.4) (line 8). Finally, each event is assigned to the class corresponding to the highest probability in the aggregated decision (line 13).

Algorithm 5 Generalized Decision Aggregation

Input: Observation matrix $\mathbf{A}$, individual decisions $\mathbf{d}_i^j$, and error threshold ϵ;
Output: The class label for each event L_i;

1: Initialize $\mathbf{x}_i^{(0)}$, $\mathbf{x}_i^{(1)}$ randomly.
2: $\tau \leftarrow 1$
3: **while** $\sqrt{\sum_{i=1}^{t} ||\mathbf{x}_i^{(\tau)} - \mathbf{x}_i^{(\tau-1)}||^2} > \epsilon$ **do**
4: **for** $j \leftarrow 1$ to n **do**
5: $w_j^{(\tau+1)} \leftarrow \log\left(\frac{\sum_{j=1}^{n}\sum_{i=1}^{t} a_{ij}||\mathbf{x}_i^{(\tau)} - \mathbf{d}_i^j||^2}{\sum_{i=1}^{t} a_{ij}||\mathbf{x}_i^{(\tau)} - \mathbf{d}_i^j||^2}\right)$
6: **end for**
7: **for** $i \leftarrow 1$ to t **do**
8: $\mathbf{x}_i^{(\tau+1)} \leftarrow \frac{\sum_{j=1}^{n} a_{ij} w_j^{(\tau+1)} \mathbf{d}_i^j}{\sum_{j=1}^{n} a_{ij} w_j^{(\tau+1)}}$
9: **end for**
10: $\tau \leftarrow \tau + 1$
11: **end while**
12: **for** $i \leftarrow 1$ to t **do**
13: **return** $L_i \leftarrow \arg\max_k x_{ik}^{(\tau)}$
14: **end for**

5.4.4 Performance Analysis

In each iteration, the GDA algorithm takes $O(mnt)$ time, where n and t represent the number of sensors and events, while $m = \max_{e_i \in \mathcal{E}} m_i$ is the maximum number of classes among all the events. Also, the convergence rate of coordinate descent method is usually linear [5] (we actually fix the number of iterations in the experiments). In practice, the number of candidate classes of the observed events and the number of sensor nodes that observe the same events are usually small. Thus, the computational complexity of the algorithm can be considered linear with respect to the number of events. Consequently, the proposed algorithm is not more expensive than the classification algorithms, and thus can be applied to any platform running classification tasks. Furthermore, since wireless/wired communication is the dominating factor of the energy consumption in distributed sensing systems, our algorithm actually saves much more energy than it consumes since it significant reduces the amount of information delivered by each sensor.

5.4.5 Example

We now walkthrough a simple example to illustrate the iterative process of the GDA algorithm. Table 5.1 provides the information of the first 5 events shown in Fig. 5.1. We omit other events' information due to space limitation. In this table, we list the sensor nodes that observe each of the 5 events, and the corresponding decision probability vectors generated by the sensors. In addition, the ground truth label of each event is given in the last column.

Table 5.1: Events

Event	Sensor Node	Decision Vector	Ground Truth
e_1	s_2 s_3	(0.1, 0.8, 0.1) (0.6, 0.2, 0.2)	1
e_2	s_1 s_3	(0.3, 0.5, 0.2) (0.1, 0.3, 0.6)	3
e_3	s_3 s_4	(0.8, 0.1, 0.1) (0.8, 0.1, 0.1)	1
e_4	s_1 s_4	(0.4, 0.3, 0.3) (0.2, 0.6, 0.2)	2
e_5	s_1 s_2 s_5	(0.5, 0.5, 0) (0.7, 0.3, 0) (0.3, 0.7, 0)	2

We apply the GDA algorithm to the sensing system in Fig. 5.1. Table 5.2 shows the aggregated decisions of the above 5 events upon initialization and convergence.

Table 5.2: Iterations of GDA

Event	Aggregated Decision	Label	Sensor	Reliability
Initialization				
e_1	(0.35,0.5,0.15)	2	s_1	1.9664
e_2	(0.2,0.4,0.4)	2	s_2	0.7999
e_3	(0.8,0.1,0.1)	1	s_3	1.6819
e_4	(0.3,0.45,0.25)	2	s_4	3.3014
e_5	(0.5, 0.5, 0)	1	s_5	1.6721
Convergence				
e_1	(0.5701,0.2358,0.1941)	1	s_1	1.6093
e_2	(0.1517,0.3517,0.4966)	3	s_2	0.2731
e_3	(0.8,0.1,0.1)	1	s_3	4.5801
e_4	(0.2505,0.5243,0.2252)	2	s_4	4.7438
e_5	(0.3712,0.6288,0)	2	s_5	3.9157

Initially, as shown in the Initialization portion of Table 5.2, the aggregated decision of each

event is set as the average of individual decisions made by the sensors that observe this event. The predicted label corresponds to the class with the highest probability. In rare cases where ties occur (e.g., event e_2 and e_5), we break them randomly.

Then, the algorithm starts to iterate, and update the values of sensor reliability and aggregated decisions repeatedly. After the algorithm converges, as shown in the Convergence portion of Table 5.2, the predicted label for each event exactly matches the ground truth. From the results, we have several observations. For example, sensor s_2 and s_3 have conflicting decisions about event e_1, and s_2 is more confident with its decision. Thus, the simple averaged decision gives a predicted label of 2, which contradicts against the ground truth. In this case, the GDA algorithm outputs the correct label, because it takes into account the reliability of individual sensors. As can be seen in Table 5.2, the reliability score of s_3 is much higher than that of s_2, so the aggregated result should favor s_3's decision.

Table 5.3: Sensor Reliability

Sensor	Correct	Incorrect	Unclear	Total	Reliability
1	0	2	1	3	1.6093
2	0	3	0	3	0.2731
3	6	0	0	6	4.5801
4	6	0	0	6	4.7438
5	3	0	0	3	3.9157

A decision is considered to be correct (or incorrect) if the class with the highest probability in the decision vector matches (or differs from) the ground truth label. If there is not a unique highest probability in a decision vector (e.g., (0.5,0.5,0)), we refer to this decision as an unclear decision. Table 5.3 shows the number of correct, incorrect, and unclear decisions made by each sensor node. As can be seen, the sensors that can make more correct decisions are assigned higher reliability scores.

5.5 Performance Evaluation

In this section, we evaluate the proposed generalized decision aggregation (GDA) framework. Experiment results on both synthetic data and a set of realistic audio recordings are presented and

discussed. We compare our GDA framework against two naïve baseline schemes as well as a state-of-the-art truth discovery approach. The experiment results show that GDA excels under various settings.

5.5.1 Synthetic Data

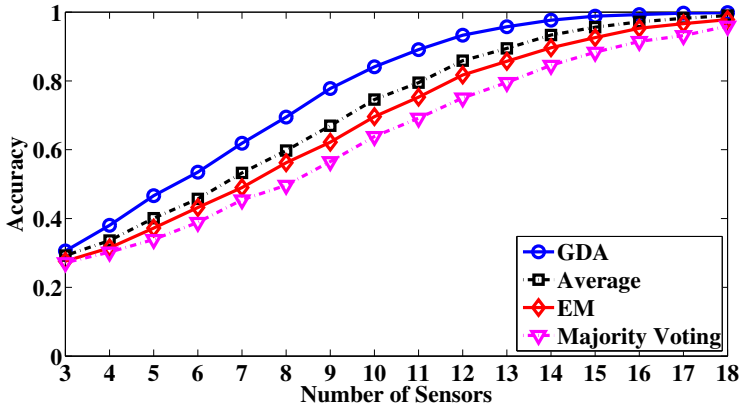
In this experiment, we simulate a sensing system where a set of events are observed and monitored by multiple sensor nodes. In particular, we randomly generate events belonging to different classes. For simplicity, we assume that each class contains the same number of events. Then, we generate sensor nodes with uniformly distributed reliability. The sensors with higher reliability are more likely to generate decision vectors whose highest probabilities correspond to the ground truth event label.

For comparison purposes, we include three baseline methods in the experiment. The first simply averages the decision probability vectors on each event and labels this event corresponding to the class with the highest probability in the averaged decision vector. The other two baselines take the discrete decision information as input. Among them, one is majority voting (i.e., count the votes for each class), the other is a state-of-the-art truth discovery approach [89–91], which uses Expectation Maximization (EM) algorithm to jointly optimize the correctness of the claims made by a group of sources and the reliability of these sources. For each individual sensor decision, we feed these two baselines with the class label that has the highest confidence in the decision vector.

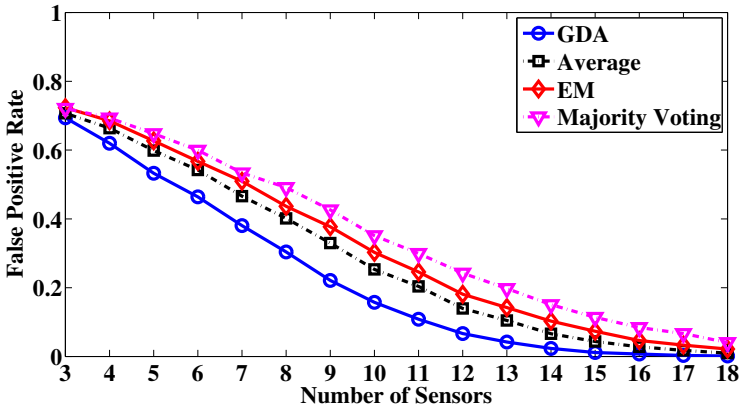
Classification Performance under Varying Number of Sensors

We first demonstrate the classification performance under varying number of sensor nodes that observe the same events. We generate 6 classes with 100 events each, where the number of observing sensors varies from 3 to 18. The experiment is repeated 10 times. We report the average results.

Figure 5.3(a) and 5.3(b) show, for all approaches, their classification accuracies (the percentage of correctly classified events, equivalent to true positive rate in this case), and false positive rates (the percentage of misclassifications of all the events that are classified to be of a particular class, then averaged among all classes), respectively. As clearly seen, our GDA framework outperforms the other approaches under any number of observing sensors in terms of both classification accura-



(a) Classification Accuracy



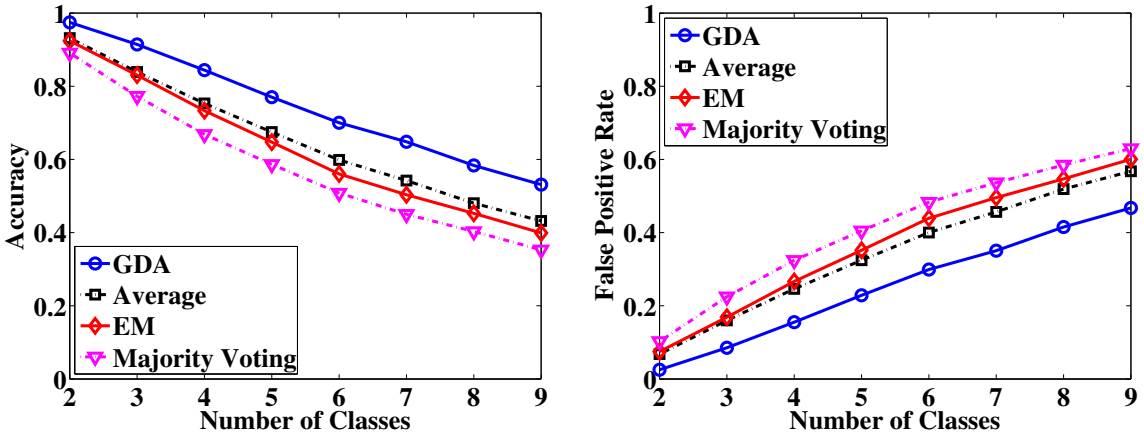
(b) False Positive Rate

Figure 5.3: Classification performance under different sensor numbers on synthetic data.

cy and false positive rate, as the classification benefits from accounting for both sensor reliability and decision confidences. On the other end of the spectrum, the majority voting yields the worst performance as it disregards useful information (sensor reliability and decision confidences) that otherwise would be useful for reaching more accurate final decisions. The EM and Average schemes takes only one factor (sensor reliability for EM, and decision confidences for average) into consideration when aggregating individual decisions, therefore they, though outperforming majority voting, still fall short compared to our GDA approach, which does utilize all useful information. One other interesting observation that we have is that all methods show similar performance when the number of sensors is either very small or quite large (e.g., 3 and 18, respectively in our experiments). This makes sense because, on one hand, when the number of sensors that observe the same events is small, it is hard to improve upon their individual poor decisions; On the other hand, as the num-

ber of sensors increases, each event is being observed by more and more diversified sensor nodes, which are more and more likely to cancel out each other's biases and errors, thus reaching better classification results. When there are a sufficiently large amount of observing sensors, even the most naïve approach (e.g. majority voting) can achieve near perfect classification accuracy.

Classification Performance under Varying Number of Classes



(a) Classification Accuracy

(b) False Positive Rate

Figure 5.4: Classification performance under different number of classes on synthetic data.

Next, we look at how GDA's classification performance compares to the other approaches with varying number of classes. The results are shown in Fig. 5.4. In this experiment, we assume that each event is observed by 10 different sensors, and each class contains 100 events. The number of classes ranges from 2 to 9.

Figure 5.4(a) and 5.4(b) show the classification accuracies and false positive rates of all approaches. As seen, our GDA approach consistently outperforms the other methods regardless of the number of classes, where the relative effectiveness of all studied approaches remains the same as that of the previous experiment. This is not surprising, as, still, the scheme that can take advantage of more information performs better. Also seen from the figures, it is clear that all approaches' classification performance degrades as the number of classes increases. This is generally expected for any classification task as the more candidate classes there are, the more confusion the classification algorithms need to comb through. We do, however, notice that as the number of classes increases, our GDA's performance degradation is slightly slower than the other approaches

in general.

Sensor Reliability

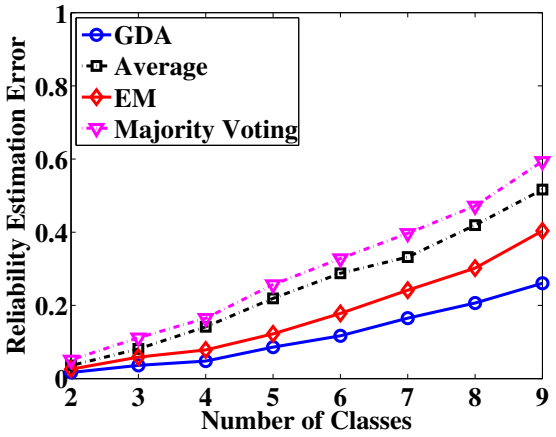


Figure 5.5: Estimation errors of sensor reliability under different number of classes.

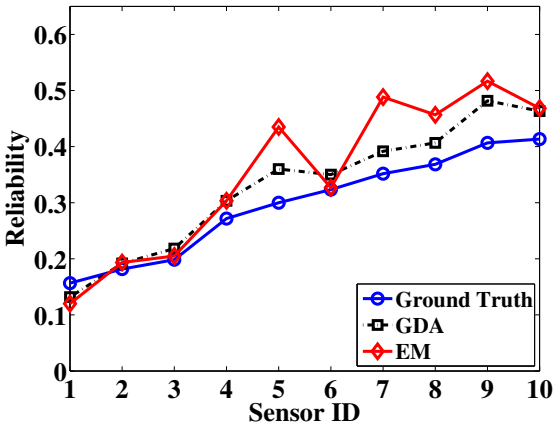


Figure 5.6: Reliability measures of 10 sensors observing the same set of events.

Using the same setting as the previous experiment, we now examine how the different schemes perform in terms of estimating sensor reliability by comparing the reliability estimation errors of our proposed GDA framework to that of the other three approaches, under varying number of classes. In particular, the reliability estimation error is computed as follows. For each individual sensor node, its ground truth reliability is defined as its standalone classification accuracy derived from comparing its individual decisions to the ground truth event labels, and its estimated reliability under a particular scheme is the classification accuracy derived from comparing its individual decisions to the aggregated decision reached by that scheme. A sensor's reliability estimation error is thus computed as the normalized distance between its estimated and ground truth reliability.

The results are shown in Figure 5.5. Similar to previous experiments' results, our proposed GDA still consistently outperforms the others. In particular, we see that the approaches that take sensor reliability into account when performing decision aggregations (i.e., GDA and EM) achieve better performance than those who do not (i.e., Average and Majority voting). Also, as the number of classes becomes larger, the estimation performance of all approaches gets poorer. Similar to the previous experiment, a higher number of classes would lengthen the distance between the aggregated

decisions and ground truth event labels, thus causing more inaccurate sensor reliability estimations. That said, we do, however, still observe that our GDA scheme shows higher robustness than the other three methods as the number of classes increases.

Figure 5.6 shows the reliability of 10 sensor nodes that observe the same set of events. For ease of illustration, we sort the sensor nodes in the increasing order of ground truth reliability. As can be seen, the ground truth reliability of these 10 sensor nodes roughly follow a uniform distribution. In Fig. 5.6, we also show the reliability as estimated by our GDA as well as the EM schemes. It is clearly seen that the estimations from our GDA framework follow more closely to the ground truth.

Convergence

Next up, we demonstrate the convergence of GDA. Specifically in the experiment we have 600 events equally distributed under 6 classes, where each event is observed by 10 different sensor nodes.

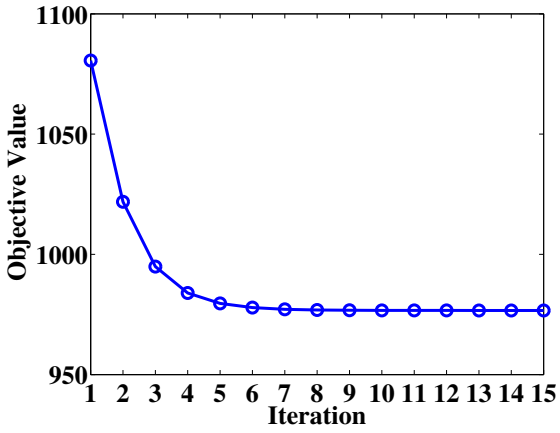


Figure 5.7: Convergence

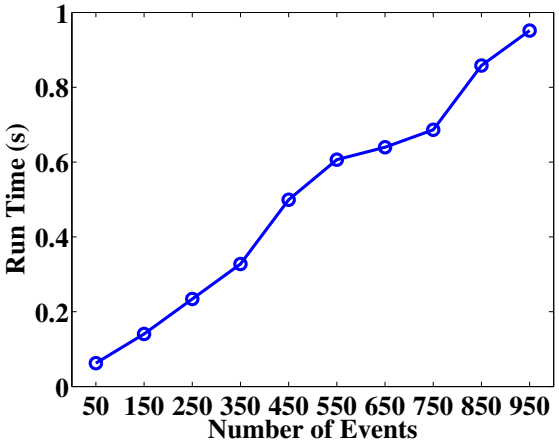


Figure 5.8: Run time

Figure 5.7 shows the evolution of the objective value (Eqn. (5.1)) of $\mathbf{P}$, which denotes the weighted summation of the distances between individual decisions and the aggregated decision. According to the GDA algorithm, the objective value is initialized based on randomly selected aggregated decisions and the resultant sensor reliability. In the subsequent iterations of the algorithm, the objective value is progressively reduced by optimizing the aggregated decisions and the

sensor reliability alternatively. As shown in Fig. 5.7, the objective value converges quickly within just a few iterations.

Complexity

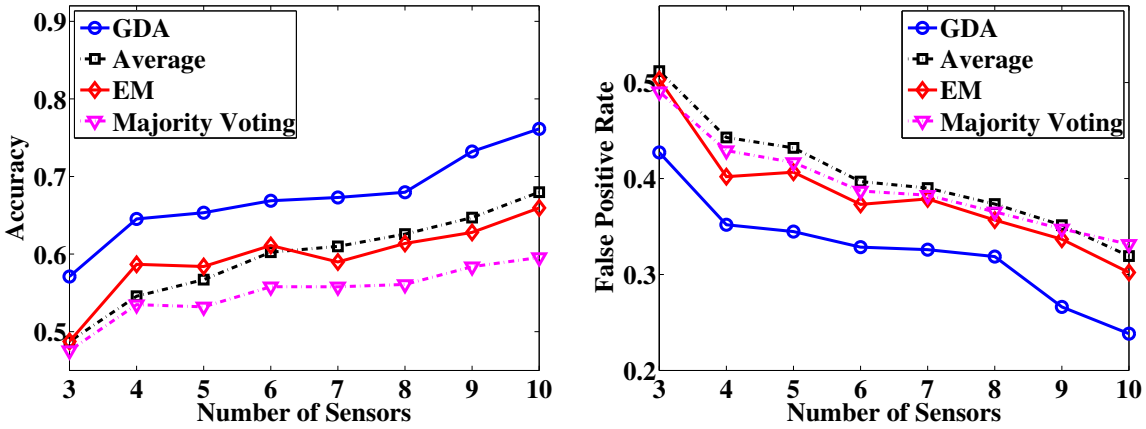
Lastly, we look at GDA's computational complexity. We demonstrate that GDA's running time is linear with respect to the number of events under practical settings where, in particular, events are equally distributed under 6 classes with each event being observed by 10 different sensor nodes.

Figure 5.8 shows the running times of GDA under different input sizes (i.e., the number of events in each class). As seen, GDA displays linear complexity with respect to the number of events. To further demonstrate this, we compute Pearson's correlation coefficient, a commonly used metric for testing linear relationship between variables. The coefficient ranges between -1 and 1, and the closer it is to 1 (or -1), the stronger the variables are positively (or negatively) linearly correlated. In our experiment, the Pearson's correlation coefficient for running time and the number of events is 0.985, indicating strong positive linear correlation.

5.5.2 Audio Data

We next shift our attention from synthetic data to realistic audio data, using which we examine the classification performance of our GDA framework as well as the three baseline approaches. The audio clips we use in this experiment include the sounds of a tank moving, a helicopter flying, and a machine gun firing, corresponding to 3 different classes. We cut the audio clips into pieces with equal time duration, and make a copy for each sensor node. We then add random noise to the sounds received by sensor nodes with various SNRs (Signal-to-Noise Ratios). Next, we extract the MFCC (Mel-Frequency Cepstral Coefficients) features from each audio clip, and feed them as the input to the classification algorithms. In this experiment, we choose random forest [7] as individual sensor node's local classifier. Random forest is a decision tree based classification algorithm that trains multiple decision trees simultaneously and has them vote for the final classification decision. Random forest can output both decision probability vectors and discrete labels (derived from decision probability vectors) that are fed to different approaches under evaluation.

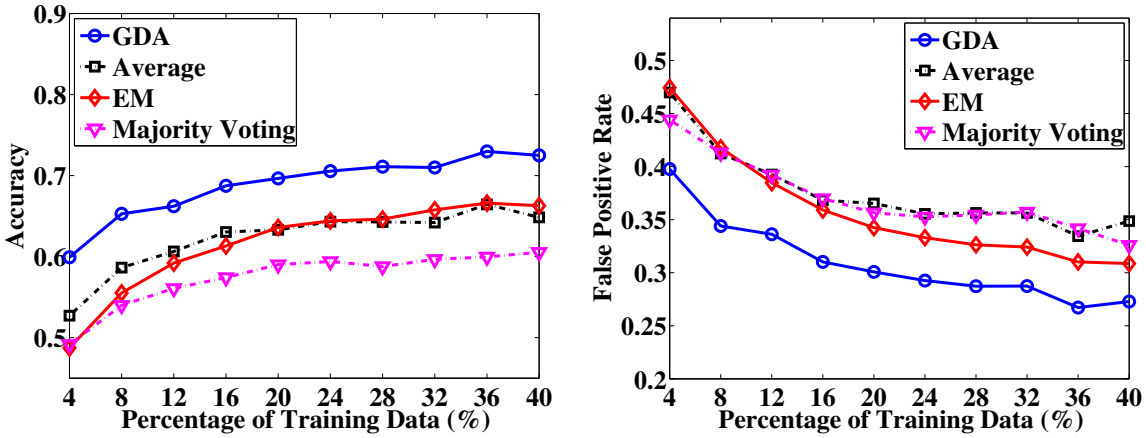
The classification result with varying number of sensor nodes is shown in Figure 5.9, which, as



(a) Classification Accuracy (b) False Positive Rate

Figure 5.9: Classification performance under different number of sensors on realistic audio data.

seen, is quite similar to that of the experiment on synthetic data shown in Figure 5.3. The curves, however, are not as smooth, due to the randomness in the audio sounds themselves. Nevertheless, we can still observe the same general performance trends as displayed in the previous experiment.



(a) Classification Accuracy (b) False Positive Rate

Figure 5.10: Classification performance under varying training data availability levels on realistic audio data.

Figure 5.10 shows the classification performance of studied approaches under varying training data availability levels. We see that the general relative classification effectiveness of all approaches remains the same as all previous experiments, with our GDA framework consistently yielding the best performance. Also, the figure shows that, for all approaches, higher training data availability lead to better classification performance, as expected.

5.6 Related Work

Classification techniques are widely used in a full range of sensing domains. For example, in habitat monitoring, sensing systems are deployed to recognize different animal species and their behaviors [9, 21, 31, 38, 60, 71]; In target surveillance and tracking, people aim to automatically differentiate varying types of targets, such as cars, tanks, humans and animals [2, 8, 30]; In environmental monitoring, weather conditions, for example, are classified based on their impacts on humans, animals, or crops [14]; In activity recognition, health care, and assisted living, sensing systems are used to monitor users' activity levels, evaluate their health statuses, and provide feedbacks accordingly [49,52,55,84,102]; In road sensing and monitoring, various systems are developed and deployed to detect and recognize traffic signs and road events [37, 42]. The list goes on. Our generalized framework proposed in this chapter can essentially be applied to all these sensing domains, addressing the decision aggregation problem by corroborating the scattered classification results and making the consolidated near-optimal final decision for the target events, and doing so in a resource efficient manner.

There are prior attempts on similar problems. For example, Su et al. [76,78] study the decision aggregation problem for remotely deployed sensing systems where very limited label information can be accessed. To tackle this problem, they let sensor nodes conduct cluster analysis, which groups data points only based on the similarity of their feature values without any training. The clustering results of sensor nodes are forwarded to the base station where they are further integrated with the limited training labels. Recently, the problem of truth discovery [88–91, 99, 100] is widely investigated by the data mining and social sensing communities. Their goal is to identify the truth from claims made by difference information sources (e.g., websites, social network users). Towards this end, TruthFinder [99, 100] adopt Bayesian analysis, where each observation's confidence is calculated as the product of its providers' reliability. Wang et al. [88] propose a Bayesian Interpretation scheme as an approximation approach to quantify the solutions produced by the TruthFinder scheme. Later, Wang et al. [89–91] further present a maximum likelihood estimator attained by the expectation maximization framework that returns the best guess regarding the correctness of each claim. These approaches suffer from a major limitation, that is, they only take as input discrete decision information. In contrast, our proposed decision aggregation framework

is able to take advantage of the confidence information of each sensor about its decision, and thus achieves higher decision accuracy.

Moreover, the proposed problem and solution in this chapter are different from the traditional data aggregation or data fusion schemes in wireless sensor networks. First, data aggregation techniques [26, 39, 48, 59, 70] do not consider sensor reliability, and usually only involve applying simple operations (e.g., mean, min, and max) directly on the raw sensory data. Thus they are different from our proposed decision aggregation framework. Second, data fusion schemes [81–83, 93] are designed to gather and combine information from multiple sensors in order to improve the accuracy of target detection and recognition. However, these work do not take into consideration the reliability of individual sensor nodes. In contrast, the proposed GDA framework jointly optimizes aggregated decisions and sensor reliability, and can be applied in more general sensing scenarios where the sensor nodes, the observed event sets, and the possible candidate classes can all be different, which can thus be combined in arbitrarily complex manners.

5.7 Conclusions

In this chapter we took a closer look at the decision aggregation problem in cyber-physical systems, where traditional approaches suffer from the limitation of only examining discrete decisions from individual sensor nodes as a way to avoid high energy cost potentially caused by excessive network transmission if raw data from sensor nodes were to be transmitted. Our proposed generalized decision aggregation (GDA) framework overcame this limitation by thoroughly accounting for and intelligently taking advantage of the decision confidence and reliability of each sensor, thus consistently achieving higher final decision accuracy over the state of the art, as we extensively demonstrated through various experiments using both synthetic and realistic data. We believe our GDA framework's superior generalizability and flexibility make it suitable for a broad spectrum of sensing domains.

Chapter 6

Conclusions and Future Work

Nowadays, as computing and communication devices become ever faster and cheaper, cyber-physical systems play an increasingly more important role in our daily lives. This line of reasoning is echoed in recent research initiatives, such as Microsoft SenseWeb project[1], Nokia SensorPlanet program[2], IBM SmarterPlanet project[3], etc. All these research efforts and solicitations envision that cyber-physical systems will become the next big driver of computing research in the foreseeable future to globally interconnect physical world and human beings.

The pervasive cyber-physical systems, though bringing people unprecedented productiveness and convenience, have incurred an explosive increase in the generation of sensory data, growing beyond our current transmission and processing capabilities. To address this challenge, in this thesis I focus my efforts on building resource-efficient systems that can intelligently allocate system resources to integrate information from a huge number of unreliable sensors so that the highest quality of information can be achieved. In this section, I conclude the thesis by first summarizing our research findings, and then discussing future research directions.

6.1 Summary

In this thesis, I develop a resource-efficient information integration toolkit consisting of four component tools. Different tools bear different design goals as well as target application scenarios, as follows.

[1] http://research.microsoft.com/en-us/projects/senseweb/
[2] http://www.sensorplanet.org/
[3] http://www.ibm.com/smarterplanet/

Bandwidth Efficient Data Aggregation

The first tool is designed for data aggregation, a traditional information integration scenario in sensor networks where only simple statistics (e.g., average, max, and min) of the sensory readings are needed. It aims at finding the optimal data collection and transmission rate for each sensor node within a distributed sensor network, such that the network throughput can be maximized, under the constraint of network bandwidth. This problem is initially formulated as a non-convex network utility maximization problem, and then transformed into a convex approximate problem. We decompose it based on the duality theory, and propose a distributed algorithm to solve the decoupled problems. The tool is built upon the proposed algorithm and demonstrates near-optimal performance from both theoretical analysis and simulations.

Energy Efficient Decision Aggregation

The second tool is proposed in order to achieve more data reduction and resource saving for decision making applications where the task is to identify the categories (classes) of the observed events or targets. By letting each sensor only report its local decision on the observed events, we can minimize data transmission and thus obtain minimum resource consumption.

The challenge of combining multiple sensors' decisions lies in the limited availability of labeled training data, due to the high cost of manual labeling in the harsh locales where a cyber-physical system is usually deployed. Without sufficiently large training sets, classification algorithms locally conducted by the sensor nodes tend to make inaccurate predictions. To address this challenge, the proposed tool lets sensor nodes employ cluster analysis that can group data points only based on the similarity of their feature values without any training. The clustering results of the sensors, together with the limited label information, are further integrated to reach the global agreement.

Decision Aggregation with Sensor Selection

The third tool is developed to serve the applications where raw data are required to be delivered as is. Given limited system resources, we have to select sensors to perform data collection and transmission based on the quality of their captured information.

To this end, we provide a quality of information based data selection and transmission service

for decision making applications in cyber-physical systems. In this scenario, the QoI of a sensor node bears two aspects, data reliability that implies the degree to which the information provided by a sensor node represents the real world, and data redundancy that represents the information overlap among different sensor nodes. Through solving an optimization problem constrained by system resources, the proposed tool can maximize the reliability of sensory data while eliminating their redundancies.

Generalized Decision Aggregation

The last tool is a generalized decision aggregation framework targeting on more general application domains. First, different from the above decision aggregation tools that only take as input discrete decision information, this generalized framework is able to take advantage of the confidence information of each sensor about its decision, and thus achieves higher decision accuracy. Second, this framework can naturally handle the scenarios where different sensor nodes observe different sets of events whose numbers of possible classes may also be different. Third, it can dynamically take advantage of ground truth label information of any availability level. This framework complements the aforementioned tools and can benefit more general sensing application paradigms.

6.2 Future Research Directions

I envision that eventually cyber-physical systems will be seamlessly integrated with the Internet. In the future, Internet will become the largest cyber-physical system. In this cyber-physical Internet, all the objects—not only the cyber-devices but also all the data carriers including human beings—can interact with each other. This will bring us great opportunities as well as new challenges. I will continue my exploration and research efforts, especially in regard to the tradeoff between QoI and system resources. Below, I list a few problems I am keenly interested in exploring next.

First, in cyber-physical systems powered by dynamic renewable energy, the reliable sensor nodes may suffer from intermittent energy depletion, and thus not be able to provide continuous data collection and distillation service. In this case, we can make use of the sensor nodes ranked as unreliable whose energy supply is sufficient. In particular, we can use multiple energy-sufficient unreliable sensor nodes to replace an energy-starving reliable node for the data collection task,

since aggregating the information of multiple unreliable nodes can often cancel out errors and reach an even more accurate result than a single reliable node. To achieve this, an energy-efficient method is needed to dynamically adjust the spending and harvesting of renewable energy in the collection, transmission, and aggregation of sensory data, such that the quantity and quality of the information delivered to the sink is optimized.

Second, in people-centric cyber-physical systems, participants consume their own resources such as computing and communicating energy. A participant would not be interested in participating in sensing tasks, unless she receives a satisfying reward to compensate her resource consumption. Therefore, it is necessary to design an effective incentive mechanism that can achieve the maximum participation given a fixed budget. A feasible solution is to reward the participants based on the data volume they have contributed [95]. However, this pricing mechanism ignores a very important factor, namely, participant reliability, which can be inferred through investigating the quality of information provided by the participant. This motivates novel pricing schemes that take into account not only the efforts a participant has spent but also the quality of information she can provide.

Additionally, there are many other research opportunities waiting for us to explore. For example, in people-centric sensing applications, participants expose themselves to potential privacy threats by sharing their personal data. Also, some participants may be malicious and try to inject false data into the system. Therefore, some mechanisms need to be developed to address the privacy and security issues. Moreover, as the generation volume of sensory data keeps increasing, traditional single-machine based back-end server will soon become a performance bottleneck, and cloud computing techniques are called for to process the sensory data in a distributed and parallel manner.

I seek to resolve these challenges through not only independent but also collaborative work on various platforms such as embedded, mobile and cloud platforms, using all kinds of analytic tools such as algorithm, data mining, machine learning, and optimization. I believe these researches can give rise to and benefit tremendously a whole new range of applications, making our world safer, smarter and better.

References

[1] S. E. Anderson, A. S. Dave, and D. Margoliash. Template-based automatic recognition of birdsong syllables from continuous recordings. *The Journal of the Acoustical Society of America*, 100(2):1209–1219, 1996.

[2] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605–634, 2004.

[3] K. C. Barr and K. Asanovic. Energy aware lossless data compression. In *Proceedings of the 1st international conference on Mobile systems, applications and services (MobiSys)*, pages 231–244. ACM, 2003.

[4] J. Beringer and E. Hüllermeier. Online clustering of parallel data streams. *Data Knowl. Eng.*, 58(2):180–204, 2006.

[5] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1995.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[7] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[8] R. R. Brooks, P. Ramanathan, and A. M. Sayeed. Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE*, 91(8):1163–1171, 2003.

[9] J. Cai, D. Ee, B. Pham, P. Roe, and J. Zhang. Sensor network for the monitoring of ecosystem: Bird species recognition. In *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP)*, pages 293–298. IEEE, 2007.

[10] M. A. Carreira-Perpinan. A review of dimension reduction techniques. *Department of Computer Science. University of Sheffield. Tech. Rep. CS-96-09*, pages 1–69, 1997.

[11] N. V. Chawla, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. Learning ensembles from bites: A scalable and accurate approach. *The Journal of Machine Learning Research*, 5:421–451, 2004.

[12] L. Chen, S. Low, M. Chiang, and J. Doyle. Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–13, 2006.

[13] S. Chen and Z. Zhang. Localized algorithm for aggregate fairness in wireless sensor networks. In *Proceedings of the 12th annual international conference on Mobile computing and networking (MobiCom)*, pages 274–285. ACM, 2006.

[14] X. Cheng, J. Xu, J. Pei, and J. Liu. Hierarchical distributed data classification inwireless sensor networks. In *Proceedings of the 6th International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pages 10–19. IEEE, 2009.

[15] J.-C. Chin, N. S. Rao, D. K. Yau, M. Shankar, Y. Yang, J. C. Hou, S. Srivathsan, and S. Iyengar. Identification of low-level point radioactive sources using a sensor network. *ACM Transactions on Sensor Networks*, 7(3):21, 2010.

[16] M. Cordina and C. J. Debono. Maximizing the lifetime of wireless sensor networks through intelligent clustering and data reduction techniques. In *Proceedings of Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2009.

[17] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, New York, 1991.

[18] Y. Cui, Y. Xue, and K. Nahrstedt. Optimal resource allocation in overlay multicast. *IEEE Transactions on Parallel and Distributed Systems*, 17(8):808–823, 2006.

[19] T. G. Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.

[20] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[21] D. Duran, D. Peng, H. Sharif, B. Chen, and D. Armstrong. Hierarchical character oriented wildlife species recognition through heterogeneous wireless sensor networks. In *Proceedings of the 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–5. IEEE, 2007.

[22] C. T. Ee and R. Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)*, pages 148–161. ACM, 2004.

[23] A. Eryilmaz and R. Srikant. Joint congestion control, routing, and mac for stability and fairness in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(8):1514–1524, 2006.

[24] S. Fagerlund. Bird species recognition using support vector machines. *EURASIP Journal on Advances in Signal Processing*, 2007(1):038637, 2007.

[25] I. Fodor. A survey of dimension reduction techniques. Technical report, Lawrence Livermore National Laboratory, 2002.

[26] J. Gao, L. Guibas, N. Milosavljevic, and J. Hershberger. Sparse data aggregation in sensor networks. In *Proceedings of the 6th international conference on Information processing in sensor networks (IPSN)*, pages 430–439. ACM, 2007.

[27] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han. Graph-based consensus maximization among multiple supervised and unsupervised models. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 585–593, 2009.

[28] A. Giridhar and P. R. Kumar. Toward a theory of in-network computation in wireless sensor networks. *IEEE Communications Magazine*, 44(4):98–107, 2006.

[29] L. Girod, M. Lukac, V. Trifa, and D. Estrin. The design and implementation of a self-calibrating distributed acoustic sensing platform. In *Proceedings of the 4th international conference on Embedded networked sensor systems (SenSys)*, pages 71–84. ACM, 2006.

[30] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A. Tirumala, Q. Cao, T. He, J. A. Stankovic, T. Abdelzaher, et al. Lightweight detection and classification for wireless sensor networks in realistic environments. In *Proceedings of the 3rd international conference on Embedded networked sensor systems (SenSys)*, pages 205–217. ACM, 2005.

[31] Y. Guo, P. Corke, G. Poulton, T. Wark, G. Bishop-Hurley, and D. Swain. Animal behaviour understanding using wireless sensor networks. In *Proceedings of the 31st IEEE Conference on Local Computer Networks (LCN)*, pages 607–614. IEEE, 2006.

[32] D. L. Hall and J. Llinas. *Multisensor data fusion*. CRC press, 2001.

[33] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, third edition, 2011.

[34] J. L. Hill and D. E. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, 2002.

[35] J.-H. Hoepman. Simple distributed weighted matchings. *arXiv preprint cs/0410047*, 2004.

[36] Y. T. Hou, Y. Shi, and H. D. Sherali. Rate allocation and network lifetime problems for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 16(2):321–334, 2008.

[37] S. Hu, L. Su, H. Liu, H. Wang, and T. Abdelzaher. Smartroad: a crowd-sourced traffic regulator detection and identification system. In *Proceedings of the 12th international conference on Information processing in sensor networks (IPSN)*, pages 331–332. ACM, 2013.

[38] W. Hu, N. Bulusu, C. Chou, S. Jha, A. Taylor, et al. The design and evaluation of a hybrid sensor network for cane-toad monitoring. In *Proceedings of 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 503–508. IEEE, 2005.

[39] W. Hu, A. Misra, and R. Shorey. Caps: energy-efficient processing of continuous aggregate queries in sensor networks. In *Proceedings of 4th International Conference on Pervasive Computing and Communications (PerCom)*, pages 10–pp. IEEE, 2006.

[40] A. Hyvarinen. Survey on independent component analysis. *Neural computing surveys*, 2(4):94–128, 1999.

[41] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. In *Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom)*, pages 66–80. ACM, 2003.

[42] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi. Traffic monitoring and accident detection at intersections. *IEEE Transactions on Intelligent Transportation Systems*, 1(2):108–118, 2000.

[43] K. Kar, S. Sarkar, and L. Tassiulas. Optimization based rate control for multirate multicast sessions. In *Proceedings of the 20th IEEE International Conference on Computer Communications (INFOCOM)*, pages 123–132. IEEE, 2001.

[44] M. Keally, G. Zhou, G. Xing, and J. Wu. Exploiting sensing diversity for confident sensing in wireless sensor networks. In *Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1719–1727. IEEE, 2011.

[45] M. Keally, G. Zhou, G. Xing, J. Wu, and A. Pyles. Pbn: towards practical activity recognition using smartphone-based body sensor networks. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 246–259. ACM, 2011.

[46] F. Kelly. Charging and rate control for elastic traffic. *European transactions on Telecommunications*, 8(1):33–37, 1997.

[47] F. P. Kelly, A. K. Maulloo, and D. K. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, 49(3):237–252, 1998.

[48] L. Krishnamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 575–578. IEEE, 2002.

[49] N. D. Lane, Y. Xu, H. Lu, S. Hu, T. Choudhury, A. T. Campbell, and F. Zhao. Enabling large-scale human activity inference on smartphones using community similarity networks (csn). In *Proceedings of the 13th international conference on Ubiquitous computing (UbiComp)*, pages 355–364. ACM, 2011.

[50] A. Lazarevic and Z. Obradovic. The distributed boosting algorithm. In *Proceedings of the 7th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 311–316. ACM, 2001.

[51] E. A. Lee. Cyber physical systems: Design challenges. In *Proceedings of the 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369. IEEE, 2008.

[52] M. Lin, N. D. Lane, M. Mohammod, X. Yang, H. Lu, G. Cardone, S. Ali, A. Doryab, E. Berke, A. T. Campbell, et al. Bewell+: multi-dimensional wellbeing monitoring with community-guided user feedback and energy optimization. In *Proceedings of the conference on Wireless Health (WH)*, page 10. ACM, 2012.

[53] X. Lin, N. Shroff, and R. Srikant. A tutorial on cross-layer optimization in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(8):1452–1463, 2006.

[54] X. Lin and N. B. Shroff. Joint rate control and scheduling in multihop wireless networks. In *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC)*, pages 1484–1489. IEEE, 2004.

[55] K. Lorincz, B.-r. Chen, G. W. Challen, A. R. Chowdhury, S. Patel, P. Bonato, and M. Welsh. Mercury: a wearable sensor network platform for high-fidelity motion analysis. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 183–196. ACM, 2009.

[56] S. H. Low and D. E. Lapsley. Optimization flow controli: basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, 1999.

[57] P. Luo, H. Xiong, K. Lü, and Z. Shi. Distributed classification in peer-to-peer networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 968–976. ACM, 2007.

[58] M. Lustig, D. Donoho, and J. M. Pauly. Sparse mri: The application of compressed sensing for rapid mr imaging. *Magnetic resonance in medicine*, 58(6):1182–1195, 2007.

[59] S. Madden, M. J. Franklin, J. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the 5th Symposium on Operating Systems Design and implementation (OSDI)*, 2002.

[60] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (WSNA)*, pages 88–97. ACM, 2002.

[61] D. Marco, E. J. Duarte-Melo, M. Liu, and D. L. Neuhoff. On the many-to-one transport capacity of a dense wireless sensor network and the compressibility of its data. In *Proceedings of the 2nd international conference on Information processing in sensor networks (IPSN)*, pages 1–16. Springer-Verlag, 2003.

[62] E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell. Darwin phones: the evolution of sensing and inference on mobile phones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services (MobiSys)*, pages 5–20. ACM, 2010.

[63] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[64] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.

[65] R. Pon, M. A. Batalin, J. Gordon, A. Kansal, D. Liu, M. Rahimi, L. Shirachi, Y. Yu, M. Hansen, W. J. Kaiser, et al. Networked infomechanical systems: a mobile embedded networked sensor platform. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 376–381. IEEE, 2005.

[66] B. Przydatek, D. Song, and A. Perrig. Sia: Secure information aggregation in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys)*, pages 255–265. ACM, 2003.

[67] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference (DAC)*, pages 731–736. ACM, 2010.

[68] K. Römer. Discovery of frequent distributed event patterns in sensor networks. In *Proceedings of the 5th European conference on Wireless sensor networks (EWSN)*, pages 106–124. Springer-Verlag, 2008.

[69] S. Santini and K. Römer. An adaptive strategy for quality-based data reduction in wireless sensor networks. In *Proceedings of the 3rd International Conference on Networked Sensing Systems (INSS)*, pages 29–36, 2006.

[70] R. Sarkar, X. Zhu, and J. Gao. Hierarchical spatial gossip for multi-resolution representations in sensor networks. In *Proceedings of the 6th international conference on Information processing in sensor networks (IPSN)*, pages 420–429. ACM, 2007.

[71] B. Sheng, Q. Li, W. Mao, and W. Jin. Outlier detection in sensor networks. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 219–228. ACM, 2007.

[72] N. Z. Shor, K. C. Kiwiel, and A. Ruszcayǹski. *Minimization methods for non-differentiable functions*. Springer-Verlag, 1985.

[73] P. Somervuo, A. Harma, and S. Fagerlund. Parametric representations of bird sounds for automatic species recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(6):2252–2263, 2006.

[74] J. A. Stankovic, I. Lee, A. Mok, and R. Rajkumar. Opportunities and obligations for physical computing systems. *Computer*, 38(11):23–31, 2005.

[75] L. Su, B. Ding, Y. Yang, T. F. Abdelzaher, G. Cao, and J. C. Hou. ocast: Optimal multicast routing protocol for wireless sensor networks. In *Proceedings of the 17th IEEE International Conference on Network Protocols (ICNP)*, pages 151–160. IEEE, 2009.

[76] L. Su, J. Gao, Y. Yang, T. F. Abdelzaher, B. Ding, and J. Han. Hierarchical aggregate classification with limited supervision for data reduction in wireless sensor networks. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 40–53. ACM, 2011.

[77] L. Su, Y. Gao, Y. Yang, and G. Cao. Towards optimal rate allocation for data aggregation in wireless sensor networks. In *Proceedings of the 12th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, page 19. ACM, 2011.

[78] L. Su, S. Hu, S. Li, F. Liang, J. Gao, T. F. Abdelzaher, and J. Han. Quality of information based data selection and transmission in wireless sensor networks. In *Proceedings of the 33rd IEEE Real-Time Systems Symposium (RTSS)*, pages 327–338. IEEE, 2012.

[79] L. Su, C. Liu, H. Song, and G. Cao. Routing in intermittently connected sensor networks. In *Proceedings of the 16th IEEE International Conference on Network Protocols (ICNP)*, pages 278–287. IEEE, 2008.

[80] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, 2005.

[81] R. Tan, G. Xing, X. Liu, J. Yao, and Z. Yuan. Adaptive calibration for fusion-based wireless sensor networks. In *Proceedings of the 29th conference on Information communications (INFOCOM)*, pages 2124–2132. IEEE, 2010.

[82] R. Tan, G. Xing, Z. Yuan, X. Liu, and J. Yao. System-level calibration for fusion-based wireless sensor networks. In *Proceedings of the 31st IEEE Real-Time Systems Symposium (RTSS)*, pages 215–224. IEEE, 2010.

[83] A. Tantawy, X. Koutsoukos, and G. Biswas. Transmission control policy design for decentralized detection in tree topology sensor networks. In *Proceedings of the 14th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2011.

[84] E. M. Tapia, S. S. Intille, and K. Larson. *Activity recognition in the home using simple and ubiquitous sensors*. Springer, 2004.

[85] Y. Tsaig and D. L. Donoho. Extensions of compressed sensing. *Signal processing*, 86(3):549–571, 2006.

[86] P.-J. Wan, K. M. Alzoubi, and O. Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. In *Proceedings of the 21st IEEE International Conference on Computer Communications (INFOCOM)*, pages 1597–1604. IEEE, 2002.

[87] P.-J. Wan, S. C.-H. Huang, L. Wang, Z. Wan, and X. Jia. Minimum-latency aggregation scheduling in multihop wireless networks. In *Proceedings of the 10th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 185–194. ACM, 2009.

[88] D. Wang, T. Abdelzaher, H. Ahmadi, J. Pasternack, D. Roth, M. Gupta, J. Han, O. Fatemieh, H. Le, and C. C. Aggarwal. On bayesian interpretation of fact-finding in information networks. In *Proceedings of the 14th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2011.

[89] D. Wang, T. Abdelzaher, L. Kaplan, and C. Aggarwal. Recursive fact-finding: A streaming approach to truth estimation in crowdsourcing applications. In *Proceedings of the 33rd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2013.

[90] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher. On truth discovery in social sensing: a maximum likelihood estimation approach. In *IPSN*, pages 233–244. ACM, 2012.

[91] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher. On truth discovery in social sensing: a maximum likelihood estimation approach. In *Proceedings of the 11th international conference on Information Processing in Sensor Networks (IPSN)*, pages 233–244. ACM, 2012.

[92] Y. Wu, S. Fahmy, and N. B. Shroff. On the construction of a maximum-lifetime data gathering tree in sensor networks: Np-completeness and approximation algorithm. In *Proceedings of the 27th IEEE International Conference on Computer Communications (INFOCOM)*, pages 356–360. IEEE, 2008.

[93] G. Xing, R. Tan, B. Liu, J. Wang, X. Jia, and C.-W. Yi. Data fusion improves the coverage of wireless sensor networks. In *Proceedings of the 15th annual international conference on Mobile computing and networking (MobiCom)*, pages 157–168. ACM, 2009.

[94] Y. Xue, Y. Cui, and K. Nahrstedt. Maximizing lifetime for data aggregation in wireless sensor networks. *Mobile Networks and Applications*, 10(6):853–864, 2005.

[95] D. Yang, G. Xue, X. Fang, and J. Tang. Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing. In *Proceedings of the 18th annual international conference on Mobile computing and networking (MobiCom)*, pages 173–184. ACM, 2012.

[96] Y. Yang, L. Su, Y. Gao, and T. F. Abdelzaher. Solarcode: utilizing erasure codes for reliable data delivery in solar-powered wireless sensor networks. In *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–5. IEEE, 2010.

[97] Y. Yang, L. Wang, D. K. Noh, H. K. Le, and T. F. Abdelzaher. Solarstore: enhancing data reliability in solar-powered storage-centric sensor networks. In *Proceedings of the 7th international conference on Mobile systems, applications, and services (MobiSys)*, pages 333–346. ACM, 2009.

[98] Y. Yang, X. Wang, S. Zhu, and G. Cao. Sdap: a secure hop-by-hop data aggregation protocol for sensor networks. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 356–367. ACM, 2006.

[99] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 1048–1052. ACM, 2007.

[100] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. *IEEE Transactions on Knowledge and Data Engineering*, 20(6):796 – 808, 2008.

[101] B. Yu, J. Li, and Y. Li. Distributed data aggregation scheduling in wireless sensor networks. In *Proceedings of the 28th IEEE International Conference on Computer Communications (INFOCOM)*, pages 2159–2167. IEEE, 2009.

[102] Z. Zeng, S. Yu, W. Shin, and J. C. Hou. Pas: A wireless-enabled, cell-phone-incorporated personal assistant system for independent and assisted living. In *Proceedings of the 28th International Conference on Distributed Computing Systems (ICDCS)*, pages 233–242. IEEE, 2008.